Metropolia

Henri Kalliosaari

Web-sovelluksen haavoittuvuuksien löytäminen DAST-järjestelmällä

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikan koulutusohjelma
Insinöörityö
3.4.2013

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Henri Kalliosaari Web-sovelluksen haavoittuvuuksien löytäminen DAST-järjestelmällä 38 sivua 3.4.2013
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja(t)	Tietoturva-asiantuntija Jani Hakala Yliopettaja Janne Salonen

Tämä opinnäytetyö tehtiin Cygate Oy:lle. Työssä tutkittiin web-sovelluksen tietoturvaa ja web-sovelluksesta etsittiin mahdollisia haavoittuvuuksia käyttäen DAST-järjestelmää. DAST tarkoittaa dynaamista sovelluksen tietoturvan testausta, ja sen tarkoituksena on emuloida sille annettujen oikeuksien avulla esimerkiksi sovelluksen normaalia käyttäjää ja tällä tavalla etsiä haavoittuvuuksia ja mahdollisia hyökkäyspisteitä annetusta rajapinnasta.

Työssä keskityttiin tutkimaan web-sovelluksen haavoittuvuuksia ja sitä, miten niiden estäminen on mahdollista hyödyntäen DAST-järjestelmän antamia tuloksia. Työssä testattiin myös IPS:n eli Intrusion Prevention Systemin kykyä estää löydettyjen haavoittuvuuksien hyväksikäyttö- ja hyökkäysyritysten huomaaminen. Työ suoritettiin virtuaaliympäristössä.

Teoriaosuudessa käytiin läpi HTTP/HTTPS-protokollia, asiakas- ja palvelinpuolen ohjelmointia ja perehdyttiin OWASP 2010 –julkaisuun, jossa mainitaan 10 yleisintä tietoturvauhkaa web-sovelluksia vastaan. Teoriaosuudessa käytiin myös läpi tietoturvalaitteistoa ja niiden toimintaa. Toteutusvaiheessa virtuaaliympäristöön asennettiin työn vaatimat virtuaalilaitteet. Työssä suoritettin kolme skannausta, jotka käytiin läpi tarkasti. Näiden perusteella tehtiin kartoitus DAST-järjestelmän toiminnasta ja tutkitiin sen luotettavuutta haavoittuvuuksien kartoittamisessa. Työssä tarkkailtiin IPS:n toimintaa ja DAST-järjestelmän laitevalmistajan ohjelmiston avulla suoritettiin haavoittuvuuksien muuntaminen IPS:n säännöiksi.

Lopuksi DAST-järjestelmän luotettavuutta tietoturvahaavoittuvuuksien havainnoinnista arvioitiin kuten myös IPS-järjestelmän kyky estää yleisimpiä tietoturvauhkia, jotka kohdistuvat web-sovellukseen. Myös DAST-järjestelmän laitevalmistajan ohjelmistolla tehty sääntökanta IPS-järjestelmään käytiin läpi ja siinä mahdolliset ongelmat havainnoitiin. DAST-järjestelmä ei ole täysin toimiva vielä, mutta sillä on selvästi arvoa tiettyjen web-sovelluksien tietoturvan kartoittamisessa.

Avainsanat	IPS, IDS, Palomuuri, Web-palvelin, Web-sovellus, Dynaaminen sovelluksen tietoturvatestaus, DAST

Metropolia

Abstract

Author(s) Title Number of Pages Date	Henri Kalliosaari Finding web application vulnerabilities with DAST-system 38 pages 3 April 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Data Networks
Instructor(s)	Jani Hakala, Security Specialist Janne Salonen, Principal Lecturer

This study was done for Cygate Oy. The objective of this Bachelor's Thesis was to evaluate DAST-system and its ability to map web-applications security level and to find out possible vulnerabilities. DAST means Dynamic Application Security Testing and its purpose is to emulate the given interface, usually basic user's, and from this interface find out the vulnerabilities and attack points.

In this study the main focus was in finding vulnerabilities from a vulnerable web-application and preventing those vulnerabilities using the results gathered from DAST-scans. Another objective for this study was to test the selected Intrusion Prevention System (IPS) and its ability to block web-application vulnerabilities and their abuse found out by the DAST-system. The work was done in a virtual environment.

The study contains two parts. The first part focuses on how web-application works, HTTP/HTTPS-protocols and clientside/serverside scripting. Also OWASP 2010 –report of most common web-application security threats was gone through. The first part also mentions common network security devices, appliances and their usage. The second part of this study focuses on the virtual lab environment. In this part the DAST-system was used to perform three scans and their results were evaluated. Based on these scans, the reliability of DAST-system was confirmed. Also the way IPS works with DAST-based rules was investigated and performed.

The final phase of the study was to review the reliability of DAST-scans and how IPS is capable of blocking vulnerabilities found by DAST against the web-application. The integration of vulnerabilities to IPS-rules was done and the problems found with it were reported. DAST-system is not a perfect way for web-application security, but it clearly holds value in securing critical web-applications and understanding their security needs.

Keywords	IPS, IDS, Firewall, Web-server, Web application, Dynamic Application Security Testing, DAST

Sisällys

1 Johdanto

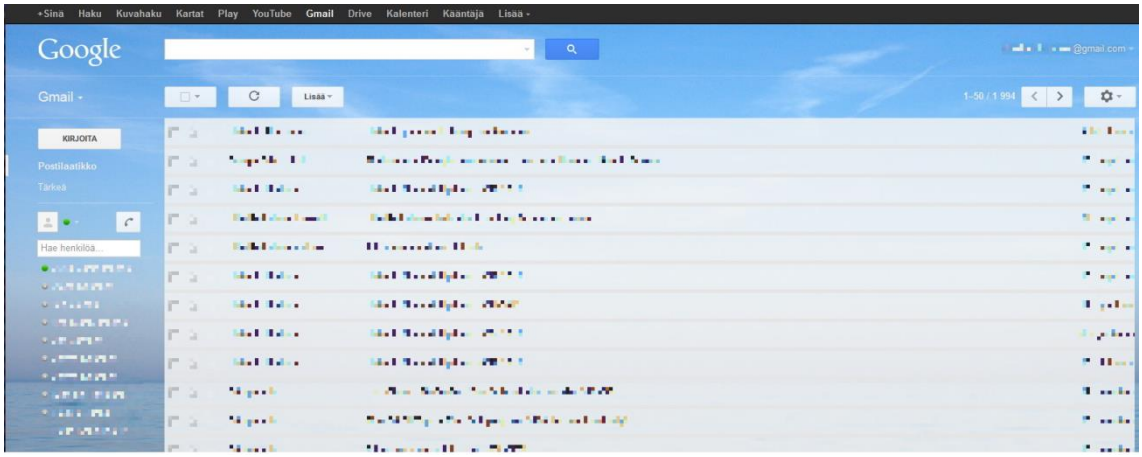
Tässä insinöörityössä tutkittiin NTOSpider DAST -järjestelmän käyttöä web-sovelluksen tietoturvan kartoittamisessa ja sen luoman haavoittuvuustietokannan kääntämistä NTODefend–ohjelmistolla joukoksi Sourcefire IPS –sääntöjä. Työ tehtiin Cygate Oy:lle, joka suunnittelee ja toteuttaa tietoverkko- ja tietoturvapalveluita asiakkailleen. Cygate Oy on osa Cygate Group Ab-konsernia ja konsernilla on yhteensä n. 700 työntekijää Ruotsissa ja Suomessa. Tämän työn tarkoituksena on tutkia DAST-järjestelmää ja sen toimintaa. Perustuen tuloksiin voidaan miettiä DAST-järjestelmän ottamista mahdollisesti Cygate Oy:n tuoteportfolioon.

Työssä tavoitteena on pystyttää virtuaalinen laboratorioympäristö, jossa sijaitsevat skannauksia suorittava palvelin, IPS-järjestelmä ja skannattava palvelin. Tässä virtuaaliympäristössä suoritetaan kolme eri skannausta. Skannauksissa selvitetään palvelimen perushaavoittuvuudet, IPS:n kyky estää haavoittuvuudet sekä niiden hyväksikäyttö ja haavoittuvuuksien muuntaminen säännöiksi Sourcefireen. DAST-järjestelmän tuoma lisäarvo jo IPS-järjestelmällä turvattuun ympäristöön on yksi työn pääkohdista.

Työ on rajattu käsittelemään web-sovelluksen tietoturvauhkia. Web-sovelluksen hierarkia ja minkä päälle se rakentuu, käydään läpi, mutta itse web-palvelimien haavoittuvuuksia ei etsitä.

2 Web-sovellus

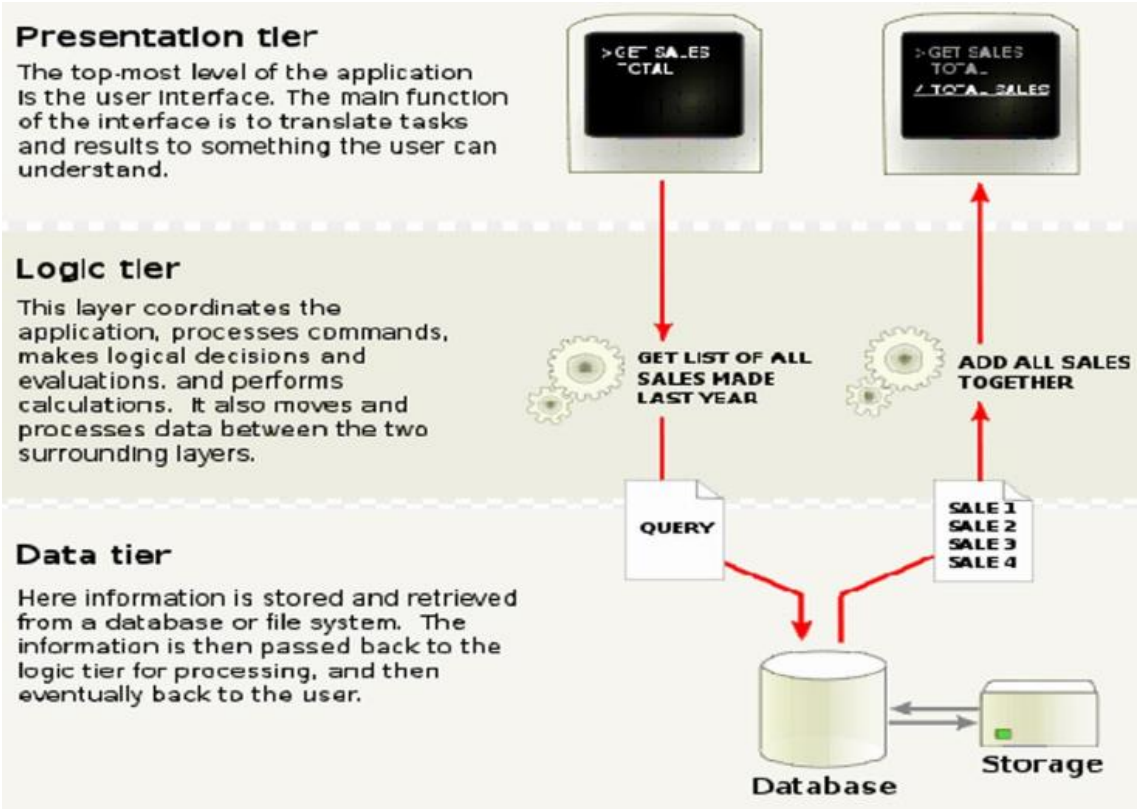
Web-sovelluksella tarkoitetaan mitä tahansa sovellusta, joka käyttää web-selainta ottaakseen yhteyttä web-palvelimeen tietoverkon yli käyttäen HTTP/HTTPS-protokollia. Web-sovellukset käyttävät asiakas-palvelin–mallia yhteydenotossa, eli järjestelmä pyytää palvelimelta tietyn palvelun suorittamista ja palvelin palauttaa pyydetyn tiedon/toiminnon järjestelmän selaimeen HTTP/HTTPS-protokollia käyttäen. Tällainen web-sovellus voi olla hyvin yksinkertainen tai erittäin monimutkainen, esimerkiksi keskustelupalsta tai tekstinkäsittelysovellus. [1.]



Kuva 1. Esimerkki web-sovelluksesta nykypäivänä, Googlen Gmail.

Web-sovelluksen yksi suurimmista eduista on sen riippumattomuus asiakkaan käyttöjärjestelmästä, jolloin web-sovelluksen kehittäjä vapautuu vastuusta sovelluksen yhteensopivuusongelmista. Riippumattomuus johtuu siitä, että itse pyyntöjä ei ajeta järjestelmässä suoraan, vaan web-selaimessa. Tällöin käyttöjärjestelmänä voi olla vaikka Windows XP tai Windows Vista. Web-sovelluksella on myös suuri etu siinä, että päivitettäessä web-palvelinta, ei asiakkaan pään järjestelmään tarvitse koskea, jolloin säästytään mahdollisesti tuhansien eri web-sovelluksen käyttäjien järjestelmien päivittämisestä. [1; 2.]

Web-sovelluksen rakenne jaetaan yleensä eri loogisiin osiin, joita kutsutaan tasoiksi. Jokaisella tasolla on oma erilainen roolinsa. On olemassa paljon erilaisia muunnelmia rakenteesta, mutta useimmiten web-sovelluksen rakenne on kolmitasoinen, jossa yleisimmin on siis kolme eri tasoa nimeltään presentaatiotaso, sovellustaso ja varastointitaso. Ensimmäisellä tasolla eli presentaatiotasolla toimii web-selain, toisella tasolla eli sovellustasolla toimii jokin sovellus käyttäen dynaamista web-sisällön ohjelmointimenetelmää, kuten ASP, ASP.NET, CGI, ColdFusion, JSP/Java, Perl, PHP, Python, Ruby jne. Viimeinen taso eli varastointitaso pitää sisällään tietokannan. Käytännössä web-selain lähettää pyynnön sovellukselle, joka suorittaa palvelun lähettämällä pyynnön edelleen tietokannalle ja palauttaa tiedon selaimelle luoden rajapinnan asiakkaan ja palvelimen välille. [3.]



Kuva 2. Web-sovelluksen kolmiosainen rakenne. [3.]

2.1 HTTP-palvelinohjelmistot Apache ja IIS

Web-sovellus pyörii web-palvelimen päällä. Web-palvelimella voidaan tarkoittaa fyysistä laitetta tai ohjelmistoa, joka toimii runkona ja ytimenä jokaiselle web-sovellukselle. Web-palvelimilla sijaitsevat siis kaikki web-sovellukset, joihin otamme yhteyttä selaimellamme käyttäessämme julkisia web-palveluita. Esimerkiksi Googlen Gmail on web-sovellus, joka pyörii jonkin web-palvelimen päällä.

Apache HTTP server on tällä hetkellä maailman suosituin palvelinohjelma. Vuodesta 2008 lähtien Apachella on ollut yli 50 prosentin osuus kaikista web-palvelimista julkisessa Internetissä. Se on täysin ilmainen ja perustuu avoimeen lähdekoodiin. Se on saatavilla monelle eri käyttöjärjestelmälle, kuten esimerkiksi Linuxille ja Windowsille. Mac OS X –käyttöjärjestelmässä se on valmiiksi integroituna. Apache on hyvin muunneltava, ja tämä on yksi sen parhaimmista ominaisuuksista. Jokainen voi muunnella Apachea omiin käyttötarpeisiinsa lisäämällä siihen eri moduuleita. Esimerkkinä voidaan käyttää vaikka mod_security:a, joka voidaan lisätä Apacheen. Tämä tuo asennettaessa

web-palvelimelle oman avoimeen lähdekoodiin perustuvan WAF:in (Web Application Firewall). Toisena esimerkkinä voidaan mainita mod_perl, joka mahdollistaa web-sovelluksessa perl-ohjelmien ajamisen ja web-sovelluksen hallinnan perl-ohjelmilla. [4.]

IIS on Microsoftin kehittämä palvelinohjelma Windows-käyttöjärjestelmiä varten. Se ei siis tue muita käyttöjärjestelmiä, vaan on täysin Microsoftin tuoteperheelle tarkoitettu. IIS ei ole avoimeen lähdekoodiin perustuva, eli sen muokkaaminen ja jakaminen on kiellettyä. IIS oli pitkään maailman toiseksi käytetyin palvelinohjelmisto, mutta on nykyään kolmannella sijalla ensimmäisenä olevan Apachen ja toisena olevan nginx:n jälkeen. IIS on hyvin paljolti käytetty yritysmaailmassa sen kattavan protokollatuen vuoksi. IIS tukee versiosta 7.5 lähtien HTTP-, HTTPS-, FTP-, FTPS-, SMTP- ja NNTP-protokollia. Kuten Apache, myös IIS on modulaarinen ja sitä voi muokata omien tarpeiden mukaan lisäämällä tietoturvamoduuleita ja HTTP-moduuleita siihen. [5.]

2.2 HTTP/HTTPS

HTTP (engl. Hypertext Transfer Protocol) on sovellustason protokolla, joka käytännössä pyörittää globaalia WWW:tä eli World Wide Web:iä. Sen tarkoituksena on mahdollistaa tiedonsiirto web-sovelluksen ja asiakkaan välillä. Se on geneerinen ja tilaton protokolla, jota voidaan käyttää myös moneen muuhun tarkoitukseen kuin pelkästään hypertekstin siirtämiseen, kuten esimerkiksi nimipalvelimissa. RFC2616 määrittää, että HTTP:tä käytettäessä pitäisi aina määrittää, mitä versiota HTTP:sta käytetään. Tällä hetkellä uusin versio HTTP:sta on 1.1. [6.]

HTTP:n määritelmä vaatii hyvän ja luotettavan siirtoprotokollan käytettäväkseen. Tämän takia yleensä HTTP-liikenne tapahtuu TCP/IP-yhteyksien päällä. Oletusporttina on 80, mutta myös muita portteja voidaan käyttää. HTTP:ta voi myös ajaa muiden siirtoprotokollien päällä (esim. UDP), mutta vaatimuksena on aina se, että siirtoprotokolla on luotettava. Tämä on oletuksena HTTP:lla eikä ilman sitä HTTP kykene toimimaan oikein. [6.]

HTTP on käytännössä pyyntö/vastausprotokolla. Asiakas eli web-selain lähettää pyynnön web-palvelimelle URI-muodossa (engl. Uniform Resource Identifier) ja protokollaversion, jota seuraa viesti sisältäen tarkennuksia/muokkaukset kuten web-selaimen tiedot. Tähän web-palvelin vastaa tilakoodillaan, viestin protokollaversiolla ja websovelluksen perustiedoilla. Telnetillä pystyy simuloimaan web-selaimen luomaa pyyntöä web-palvelimelle ja siitä generoituvaa vastausta. [6.]

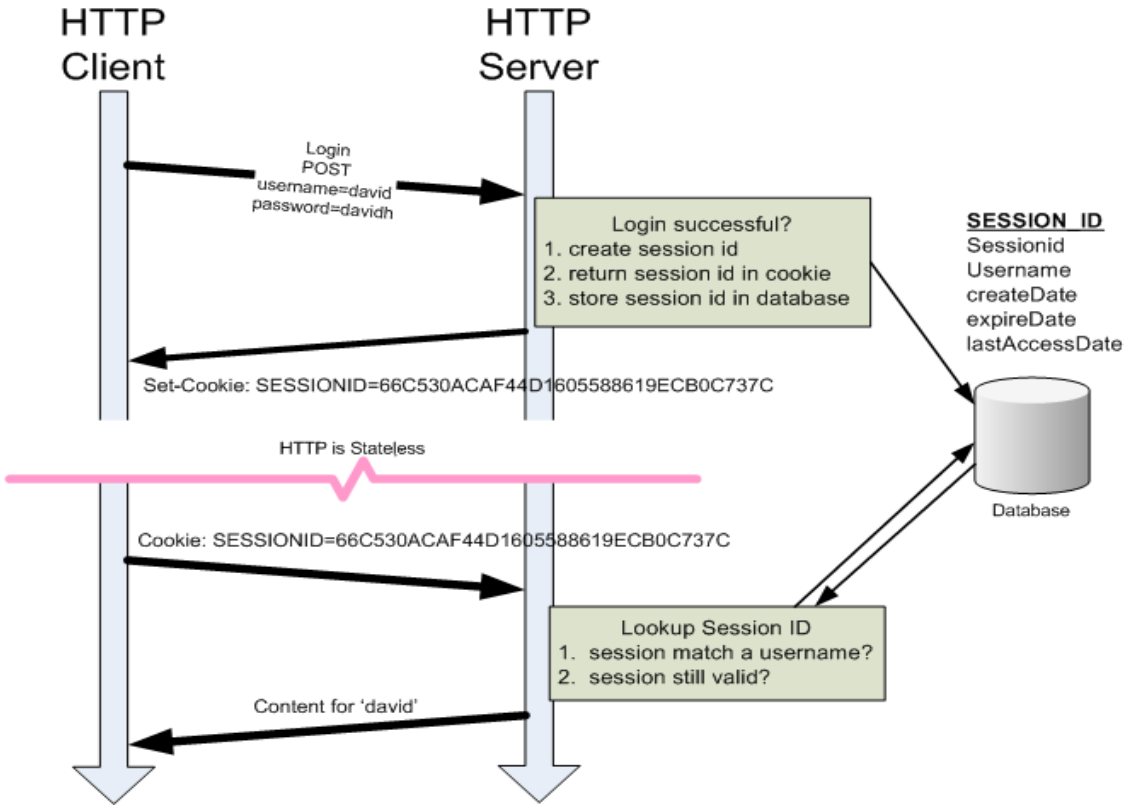
```
root@bt:~# telnet www.cygategroup.com 80
Trying 193.45.143.170...
Connected to www.cygategroup.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.cygategroup.com
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)

HTTP/1.1 200 OK
Date: Thu, 15 Nov 2012 12:31:00 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Set-Cookie: ASP.NET_SessionId=jbmqso452vgcn3usbareqt45; path=/; HttpOnly
Cache-Control: private
Expires: Thu, 15 Nov 2012 12:31:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 29978
```

Kuva 3. Telnet-yhteys Cygaten sivuille. Headeriin pystyy määrittämään enemmänkin vaihtoehtoja, mutta niitä ei ole käytetty tässä esimerkissä selvyyden vuoksi.

Koska HTTP on tilaton, luotiin tätä varten evästeet (engl. Cookie). Evästeen tarkoituksena on lähettää web-palvelimelta pieni datamäärä asiakkaan selaimeen, joka jää selaimen muistiin. Web-palvelin voi pyytää evästettä käyttäjän http-pyynnön yhteydessä myöhemmin. Eväste voi kertoa palvelimelle esimerkiksi, kuka olet ja oletko jo kirjautunut websovellukseen. Sillä voidaan myös seurata käyttäjän selailua websovelluksessa eli mitä sivuja käyttäjä avaa. Evästeen web-palvelimelta käyttäjä saa yleensä ensimmäisellä kerralla, kun selain ottaa yhteyttä web-sovellukseen ja jokin käyttäjän pyytämä sivu tai tieto palautetaan käyttäjän selaimeen. Eväste saapuu tämän datan mukana set-cookie–kentässä ja tallentuu selaimen muistiin (kts. kuvio 3). Tämän jälkeen jokaisen http-pyynnön yhteydessä selain lähettää evästeen sille web-palvelimelle, joka evästeen on selaimelle alun perin lähettänyt. Evästeitä voi olla useita per verkkotunnus

(engl. domain), eli yhden web-sivuston selailusta selaimeen voi tallentua montakin eri evästettä, joita palvelin voi tarvittaessa käyttää. [7.]

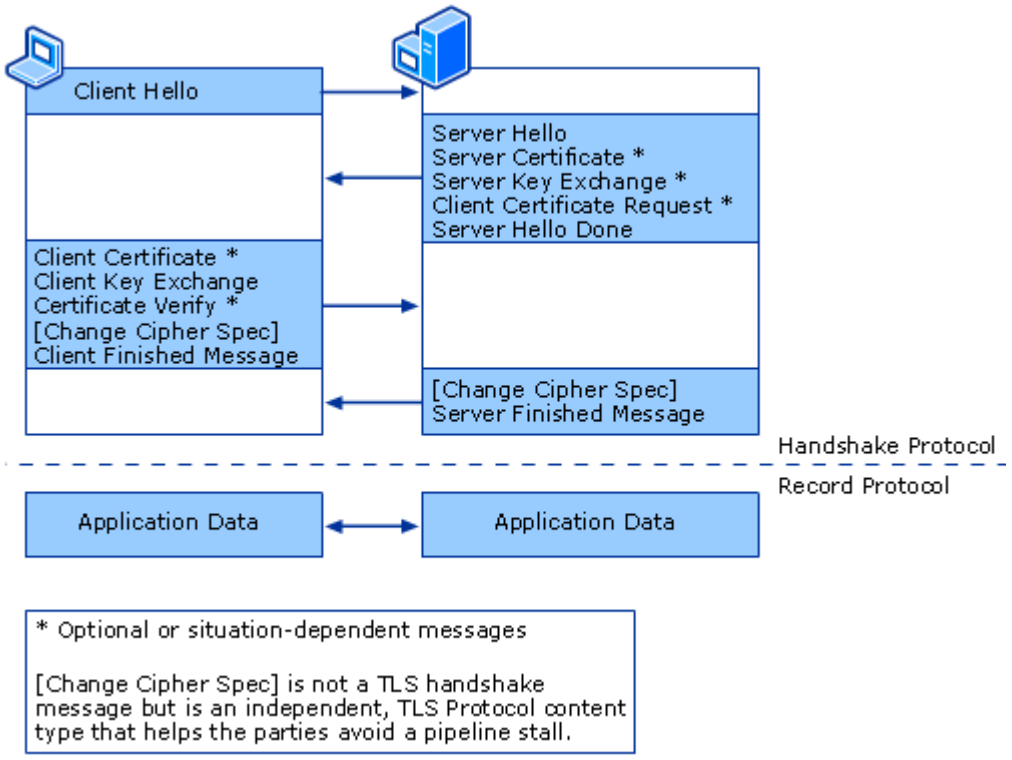
Evästeitä on kahdenlaista mallia. Nämä mallit ovat joko pysyviä evästeitä tai sessiokohtaisia evästeitä. Sessiokohtaiset evästeet poistetaan käyttäjän selaimen muistista yleensä session päätyttyä ja ne poistetaan myös web-palvelimen päässä. Pysyvät evästeet ovat valideja yleensä tietyn ajan verran tai jopa niinkauan kunnes selaimen muistista ne tuhotaan itse. [7; 8.]



Kuva 4. Esimerkki session ylläpitämisestä ja käyttäjän autentikoitumisesta palvelimelle. Palvelin luo evästeen ja lähettää sen käyttäjälle kirjautumisen onnistuttua. Jokainen http-pyyntö palauttaa palvelimelle evästeen, jotta palvelin näkee pyynnön tulevan autentikoituneelta käyttäjältä. [8.]

Nykyään on hyvin paljon web-sovelluksia, jotka käsittelevät ihmisten henkilökohtaisia asioita, kuten esimerkiksi verkkopankkiliikennettä. Yksityisyyttä ja tietoturvaa vaativat web-sovellukset käyttävät HTTPS-protokollaa, eli salattua HTTP-liikennettä. HTTPS on hyvin samankaltainen HTTP:n kanssa. Konseptina se tarkoittaa HTTP:n käyttämistä TLS-protokollan päällä, kun normaalisti HTTP:ta käytetään suoraan TCP-protokollan

päällä. Asiakas, joka toimii web-sovelluksen käyttäjänä, toimii TLS-asiakkaana myös aloittaen yhteyden muodostamisen. Se lähettää TLS-aloituskättely-paketin web-sovelluksen palvelimelle, jolloin TLS-kättely alkaa. Kun tämä kättely on valmis, voidaan HTTP-pyyntö lähettää. Kaikki HTTP-liikenne lähetetään TLS-protokollan sovellusdatana. Asiakas tietää, että web-sovelluksen palvelin on varmasti se, joka väittää olevansa, koska palvelin tarjoaa varmenteen, johon asiakas luottaa. Varmenne on CA:n (Certificate Authority) myöntämä takaus siitä, että yritys tai palvelin on juuri se, joka väittää olevansa. [9; 10.]



Kuva 5. TLS-kättely, autentikoidun ja salatun kanavan muodostaminen asiakkaan ja palvelimen välillä. Esimerkiksi ”Application Data” voi olla HTTP-liikennettä. [9.]

HTTPS:n käyttämä TLS-protokolla tarjoaa turvallisen menetelmän yhteyden sulkemiselle. Kun validi sulkemispyyntö vastaanotetaan, TLS pystyy takaamaan, että dataa ei enää liiku kyseisen yhteyden päällä. Asiakas voi sulkea yhteyden milloin tahansa välittömästi, kunhan lähettää ensiksi sulkemispyynnön palvelimelle. Jos tämä yhteys suljetaan ennen palvelimen sulkemispyynnön saapumista takaisin asiakkaalle, yhteys sul-

keutuu vajaasti (engl. Incomplete closure). Tämä jää merkkinnäksi palvelimen puolelle. Tällöin yhteyden voi palauttaa asiakkaan kautta. [10.]

HTTPS käyttää yleisesti TCP-protokollaa, mutta voi käyttää myös muita siirtoprotokollia, kuten HTTP tekee. Oletuksena TCP-yhteyksissä porttina toimii 443, mutta myös muita portteja voidaan käyttää. Web-selaimessa URI-formaattia on muutettu HTTP:sta eroavaksi siten, että protokollan tunnistuskenttään merkitään https:// eikä http://. [10.]

2.3 Asiakaspuolen koodi

Jotta koko web-sovelluksen toiminta ei olisi vain web-palvelimesta kiinni, vaan siihen saataisiin myös asiakaspuolen järjestelmä tekemään osa työstä, käytetään asiakaspuolen koodia. Asiakaspuolen ohjelmoinnilla tarkoitetaan siis HTML-dokumentissa sisällytettyä ohjelmaa, jonka web-sovellus suorittaa asiakkaan selaimessa. Asiakaspuolen koodin suorittaminen kuormittaa ja käyttää asiakkaan järjestelmää, tällöin voidaan säästää esimerkiksi web-palvelimen prosessorin ja muistin käyttöä. Asiakaspuolen koodi ja suoritettavat ohjelmat myös valvovat esimerkiksi täytettävien kohtien oikeanlaisuutta. [11.]

Asiakaspuolen koodia käytetään eniten web-sovelluksen käyttömukavuuden maksimoimiseen ja esteettisyyteen. Selain suorittaa tietyn HTML-dokumentin ohjelmat asiakkaan järjestelmässä ja näyttää tulosteen erilaisena. Tällä tavalla pystytään luomaan erilaisia efektejä kuten esimerkiksi kello web-sovellukseen. Loppujen lopuksi tarkoituksena on kuitenkin käyttöliittymän mukavuus web-sovelluksessa. [11.]

Esimerkkejä asiakaspuolen ohjelmointikielistä on esimerkiksi HTML, Dynaaminen HTML, XML, JavaScript ja XHTML. Näistä eniten käytettyjä ovat HTML ja JavaScript, yleensä JavaScript-ohjelmat on sisällytetty (ohjelmoitu) HTML-dokumenttiin tai ne voivat olla myös erillisenä tiedostona, josta ohjelma haetaan. [11.]

2.4 Palvelinpuolen koodi

Palvelinpuolen ohjelmoinnilla tarkoitetaan web-palvelimessa tapahtuvaa ohjelmointia, joka suoritetaan esimerkiksi asiakkaan pyyntöä varten. Tällöin asiakkaan selaimeen palautetaan valmis html-dokumentti. Yleisiä web-palvelimen ohjelmointikieliä on esimerkiksi ASP, PHP ja Perl. [12; 13 s.178.]

Esimerkkinä voidaan käyttää PHP:ta. Kun asiakas syöttää esimerkiksi Googlen hakukoneeseen tietyn sanan, lähtee tämä pyyntö web-palvelimelle. Tämän jälkeen PHP-ohjelma tekee haun tietokannasta tälle sanalle, ja tietokanta palauttaa datan web-palvelimelle. Tämän jälkeen web-palvelin palauttaa datan HTML-dokumenttina asiakkaan selaimeen. Asiakas ei siis missään vaiheessa näe PHP:ta, vaan PHP toimii web-palvelimessa itsessään ja suorittaa tietyn ohjelman, tässä tapauksessa haun, asiakkaan pyynnöllle tietokantaan. Asiakas näkee ainoastaan PHP-ohjelman tulosteen. [12.]

3 Web-sovelluksen tietoturva

3.1 Yleisimmät tietoturvauhat web-sovellusta vastaan

Ennen sovellustason suojausta on tärkeää pystyä suojaamaan itse web-palvelin tietyiltä alemman OSI-mallin tasojen haavoittuvuuksilta. Näitä ovat esimerkiksi yleisesti tarpeettomat portit, pääsy web-palvelimen hallintaosoitteeseen ja erilaiset palvelunestohyökkäykset. Myös web-palvelimen koodissa voi olla tiettyjä haavoittuvuuksia, jotka antavat niiden hyväksikäyttäjälle mahdollisuuden web-sovelluksen toiminnan vaikuttamiseen. [14.]

OWASP julkaisi vuonna 2010 seuraavan top10–listauksen suurimmista tietoturvauhkista web-sovelluksia vastaan. Listassa eivät ole kaikki mahdolliset uhat, mutta lista antaa hyvän kuvan siitä, mitä vastaan tulee suojautua, kun kyseessä on web-sovellus. [15.]

Injektiot

Injektioilla tarkoitetaan SQL, LDAP, HTTP header injektiota ja käyttöjärjestelmän komentoinjektioita. Nämä ovat mahdollisia silloin, kun tuntematonta dataa, kuten kysely

lähetetään web-sovellukselle. Tällä tavalla voidaan saada pääsy web-sovellukseen ja mahdollisesti muokkaamaan ja näkemään dataa. [15.]

Cross site scripting eli XSS

XSS-haavoittuvuutta hyväksikäyttäen hyökkääjä voi lähettää asiakkaan selaimeen dataa web-sovelluksen kautta, joka ei ole validia. Tällä tavoin hyökkääjä voi saada käyttäjän istunnon haltuunsa, varastaa evästeet tai tehdä jopa uudelleenohjauksia vaarallisille sivustoille. [15.]

Huono autentikoinnin- ja istunnonhallinta

Huonolla autentikoinnin- ja istunnonhallinnalla tarkoitetaan sitä, että käyttäjäksi voidaan naamioitua puutteellisen suojauksen vuoksi autentikointiprosessissa. Yleisimpiä syitä tähän on se, kun istunnon tunnus on näkyvissä muille eikä SSL/TLS suojausta ole käytössä. [15.]

Insecure Direct Object References

Turvattomasti määritellyt sisäiset objektit web-sovelluksessa ovat myös yksi suurimmista haavoittuvuuksista. Haavoittuvuutta käyttävä henkilö voi esimerkiksi vaihtaa osoitinrivinsä URL:ia ja täten saada pääsyn tiedostoon, johon hänellä ei normaalisti ole oikeutta. Myös avoimet uudelleenohjaukset ovat riski, koska hyökkääjä voi käyttää näitä phishing-tarkoituksessa. [15.]

Cross Site Request Forgery (CSRF)

Tässä hyökkääjä luo haitallista koodia huijatakseen käyttäjää täyttämään väärennetyn lomakkeen. Tässä lomakkeessa voi olla melkeinpä mitä tahansa, ja tämä on hyvin yleinen pankkihuijauksissa, joissa käyttäjän tililtä viedään rahaa. [15.]

Väärin konfiguroitu tietoturva

Tässä hyökkääjä voi esimerkiksi skannata kyseistä web-sovellusta ja huomata mahdolliset haavoittuvuudet siellä, jotka järjestelmänvalvoja on luullut estäneensä, mutta esimerkiksi vahingossa on jättänyt tekemättä. [15.]

URL:eihin pääsyn rajoittamisen puute

Hyökkäys tapahtuu siten, että sisäänkirjautunut käyttäjä pääsee katselemaan muita sivuja suoraan vaihtamalla URL:ia suoraan osoitinrivillä. Tämän hyökkäyksen kohteena on yleensä web-sovelluksen hallintasivut. [15.]

Ei-validoidut uudelleenohjaukset ja edelleenohjaukset

Ei-validoitu parametri antaa hyökkääjälle mahdollisuuden valita kohdesivun, johon he haluavat asiakkaan lähettää antamaan yksityisiä tietoja. Asiakkaat luottavat näihin yleensä, koska linkki on validille sivulle. [15.]

Turvaton datan tallennus tekstinä eikä salattuna

Yleisin syy tähän tietoturvauhkaan on se, että data, jonka pitäisi olla salattua, ei ole. Tämä voi johtua huonoista salausalgoritmeista tai vahvojen salausalgoritmien huonosta käytöstä. [15.]

Puuttellinen liikenteen salaus verkossa

Tämä haavoittuvuus on yleisin silloin, kun jokin tietty sivusto ei käytä SSL/TLS-salausta sivuille, jotka vaativat autentikoitumista. Tällöin hyökkääjä voi monitoroida tietoverkossa kulkevaa liikennettä ja saada selville autentikoidun käyttäjän istunnon evästeen. Myös vanhentuneet ja muuten puutteelliset SSL-sertifikaatit voivat saada käyttäjät hyväksymään mahdollisesti vaarallisen sivuston sertifikaatin. [15.]

3.2 Palomuuri (ensimmäinen ja toinen sukupolvi)

Palomuurikonseptilla tarkoitetaan fyysistä tietoturvalaitetta tai ohjelmistoa, jonka tehtävänä on sallia ja estää tietoliikennettä ja toimia välityspalvelimena tietoliikenteelle siten kuin organisaation tietoturvapolitiikassa on määritelty. Palomuurikonseptilla on yleistä kontrolloida tietoliikennettä eri alueiden välillä perustuen alueiden luotettavuuteen. Luotettavuus määritetään alueen tietoturvallisuuden perusteella. Internet yleensä määritellään ei-luotetuksi alueeksi, ja esimerkiksi yrityksen sisäverkko taas määritellään korkean luotettavuuden alueeksi. Tällöin yrityksen sisäverkosta aloitettu liikenne yleensä sallitaan ja Internetistä aloitettavat yhteydet oletuksena yrityksen sisäverkkoon estetään. Palomuurin tavoitteena on siis luoda hallitut rajapinnat eri luotettavuusalueiden

välillä, jolloin yrityksen tietoturvapolitiikka voidaan toteuttaa ja samalla yrityksen toiminnan vaatimat yhteydet voidaan sallia hallitusti. Kun yritys tarjoaa Internetiin web-palveluja, web-palvelut sisältävä palvelin sijoitetaan yleensä demilitarisoidulle alueelle eli DMZ:lle. Demilitarisoidulla alueella sijaitsevat palvelimet sisältävät dataa, johon täytyy olla pääsy yrityksen ulkopuolelta. Demilitarisoidun alueen luotettavuus on määritetty yleensä Internetin ja yrityksen sisäverkon välille. [16.]

Ensimmäisen sukupolven palomuurit olivat yksinkertaisia pakettisuodattimia. Sääntöjensä perusteella nämä palomuurit sallivat, estivät tai hylkäsivät niille tulevaa liikennettä. Liikenteen hylkääminen tarkoittaa sitä, että virheilmoitus paketin estämisestä lähetettiin paketin lähdeosoitteelle. Palomuuri tarkistaa usein paketin lähde- ja kohdeosoitteen ja protokollan. Kohdeportti voidaan tarkistaa TCP- ja UDP-liikenteellä. Yksinkertainen pakettisuodatin ei välitä, onko liikenne meno- vai paluuliikennettä, eli se ei varastoi tietoa yhteystilasta. Tämä tarkoittaa sitä, että myös paluuliikenne pitää sallia palomuurilla. [16.]

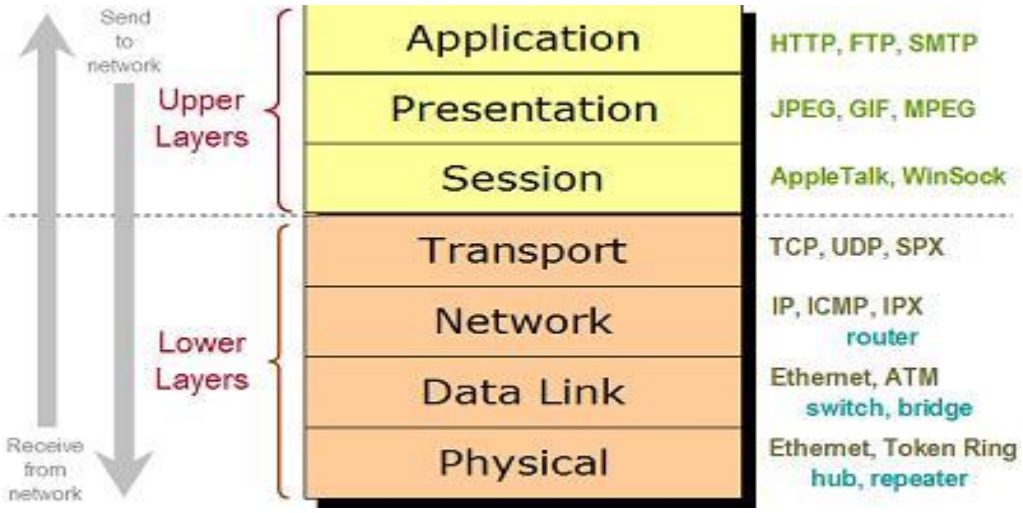
Toisen sukupolven palomuuriteknologialla tarkoitetaan yleensä tilallista palomuuria. Tilallinen palomuuri pystyy säilyttämään tiedot yhteyden tilasta muodostamalla session ja vertaamalla saapuvia paketteja voi päätellä, onko palomuurille saapuvat paketit meno- vai paluuliikennettä ja kuuluvatko ne johonkin tiettyyn sessioon. Myös yksi suuri ero tilattomaan palomuuriin eli pakettisuodattimeen on se, että sääntökannassa voidaan ottaa huomioon tulevan liikenteen tila. Tämä mahdollistaa tarkemman sääntökannan ja sääntöjen luomisen. Tilallinen palomuuri pystyy myös estämään tietynlaisia tietoliikennehyökkäyksiä, jotka hyväksikäyttävät olemassaolevia yhteyksiä, tai palvelunestohyökkäyksiä kuten SYN(tcp)-tulvahyökkäystä. Tilallinen palomuuri voi myös havaita ja estää porttiskannauksia. [16.]

3.3 Sovelluspalomuuri (kolmas sukupolvi)

Sovelluspalomuuri eli kolmannen sukupolven palomuuren isänä voidaan pitää Marcus Ranumia, jonka kehitys sovelluspalomuuriteknologian parissa synnytti ensimmäisen kaupallisen sovelluspalomuuren. Tämän tuotteen julkaisi yritys nimeltä DEC (Digital Equipment Corporation), ja tuotteen nimeksi tuli SEAL (Secure External Access Link). Myöhemmin SEAL:in nimeksi vaihdettiin AltaVista Firewall. Voidaan sanoa, että sovel-

luspalomuuren läpimurto tapahtui kesäkuun 13. päivä vuonna 1991, kun DEC sai suuren tilauspyynnön SEAL:ia Yhdysvaltojen itärannikolla sijaitsevalta kemikaalialalla toimivalta yritykseltä. [17.]

Applikaatiopalomuurin suurimpina etuina on sen tietoisuus yleisimpien protokollien toiminnasta. Näitä protokollia on esimerkiksi FTP, DNS ja HTTP. Applikaatiopalomuuri huomaa, jos ei-toivottuja protokollia yritetään salaa ajaa poikkeavasta portista tai jos jotain tiettyä protokollaa yritetään hyväksikäyttää jotenkin haitallisella tavalla. Se ei tarkasta ainoastaan paketteja verkkotasolla, vaan tutkii ja skannaa niitä täten nähden niiden sisällön. [17; 18.]



Kuva 6. Sovelluspalomuuri ymmärtää ylimpiä osi-mallin tasoja, josta se saa myös nimensä. [19.]

Sovelluspalomuuri asetetaan yleensä isommissa ympäristöissä web-palvelimen ja käyttäjän välille kun sillä halutaan suojata web-sovellusta, jolloin se tarkistaa näiden kahden pisteen välillä kulkevan liikenteen. Tämä tarkistettu liikenne tutkitaan palomuurilla ja sen sisältöä verrataan esimerkiksi sääntökantaan tai valmistajalta tulleeseen tietokantaan. Tämän perusteella palomuuri tekee päätöksensä tiedon laadusta ja oikeellisuudesta. Esimerkiksi käyttäjän lähettämää dataa voidaan määrätä tutkittavaksi kiellettyjen SQL-komentojen varalta palomuurin sääntökannassa. Näillä komennoilla voidaan suorittaa SQL-injektiohyökkäyksiä applikaatiopalvelinta vastaan tietyissä tilanteissa.

Kun palomuuri huomaa tämän, se päättää datan kohtalosta pohjaten päätöksensä sääntökannassa määritettyyn toimintaan. Nämä toiminnat voivat olla esimerkiksi liikenteen estäminen tai liikenteen siirtäminen ”hunajapurkille”. [18.]

3.4 IDS/IPS

IDS tulee sanoista Intrusion Detection System. Sillä tarkoitetaan järjestelmää, joka monitoroi ja skannaa liikennettä tai järjestelmän tilaa tietoteknisessä järjestelmässä tai tietoliikenneverkossa. IDS koostuu eri loogisista osista, jotka voivat sijaita eri fyysisillä järjestelmillä. Nämä osat ovat sensori/sensorit, tietokanta/hälytyskoneisto ja käyttöliittymä. Tämä voi kuitenkin muuttua riippuen IDS:stä. [20.]

IDS asetetaan yleensä datapolun ulkopuolelle ja sijaitsee SPAN-portin (Switched Port Analyzer) takana. Kun IDS tarkastelee liikennettä tai järjestelmän tilaa, se vertaa sitä sääntökantaansa ja sääntökannan pohjalta luo mahdollisen tapahtumahälytyksen. Sääntökannassa on määritelty yrityksen tietoturvapolitiikan mukaisesti yleinen ja sallittu liikenne ja liikenne, joka voi mahdollisesti olla tietoturvariski. Yleisimmin tapahtumahälytys luodaan esimerkiksi, kun IDS huomaa haittaohjelmia, mahdollisia etäyhteysaukkoja, oikeuksiaan väärinkäyttäviä käyttäjiä tai muita tietoturvatapahtumia. Vaikka moni tietoturvatapahtuma onkin luonteeltaan haitallinen, kaikki eivät ole. Esimerkiksi käyttäjä saattaa yrittää ottaa yhteyttää laitteeseen, jonka IP-osoitteen käyttäjä on kirjoittanut väärin. Jos käyttäjällä ei ole tähän järjestelmään pääsyä, IDS saattaa luoda tapahtumahälytyksen tästä. IPS ja IDS molemmat pyrkivät siis verkossa olevien haavoittuvuuksien hyväksikäytön estämiseen ja niiden raportoimiseen. [20; 21.]

IPS eli Intrusion Prevention System toimii samalla tavalla kuin IDS, mutta se sijoitetaan yleensä tietoliikenteen väliin. Se tarkkailee tietoverkossa liikkuvaa dataa ja vertaa sitä ”normaaliin” liikenteensä. Tämä normaaliliikenne on määritetty järjestelmän sääntökannassa juuri kuten IDS:ssä. Mikä erottaa IPS:n IDS:stä, on sen kyky tehdä enemmän kuin ainoastaan kirjata tapahtuma ylös ja luoda hälytys. IPS:llä on mahdollisuus puuttua ja reagoida siihen, mitä se huomaa verkossa tai järjestelmässä tapahtuvan. Tämän takia IPS on yleisesti halutumpi kuin IDS. [20; 21.]

IPS:ssä on myös riskinsä. IPS yleensä määritetään estämään liikenne, jonka se uskoo olevan haitallista. Ongelmaksi voi muodostua se, että IPS:llä ei ole kykyä ymmärtää web-sovelluksen protokollan toimintaperiaatteita tai logiikkaa. Tällöin IPS ei aina voi varmaksi tietää, onko jokin pyyntö aiheellinen vai aiheeton ja haitallinen applikaatiotasolla. Koska yleisesti aiheellinen liikenne ei koskaan saisi joutua estetyksi, joudutaan IPS:än sääntökantaa pitämään tietyllä tavalla löyhänä, jotta tuotantoliikenne toimii. [21.]

Web-sovelluksien määrä nykyään on valtaisa. Osa web-sovelluksista on kaupallisia ja osa ei, joten signatuuritietokantaa on hankala ylläpitää. Tästä aiheutuu suuri määrä erilaisia haavoittuvuuksia, joita voidaan hyväksikäyttää. IPS-järjestelmät eivät voi tehokkaasti kattaa kaikkia mahdollisia haavoittuvuuksia ja saattavat alkaa tuottaa paljon ns. false positiveja eli vääriä hälytyksiä. Aiheellisen liikenteen estämisen ja väärien hälytyksien tutkimisen lisäksi ongelmana on aiheellisten hälytyksien huomaaminen. Väärien hälytyksien suurta määrää kutsutaan meluksi. [21.]

3.5 WAF

WAF eli Web Application Firewall on tarkoitettu suojelemaan web-sovelluksia ja web-palvelimia hyökkäyksiltä, joita IPS ei kykene estämään. WAF:in toimintaperiaate on samanlainen kuin IPS:llä. WAF-järjestelmä tutkii liikennettä, joka saapuu ja lähtee web-sovellukselta ja -palvelimelta ja se kykenee analysoimaan applikaatiotason logiikkaa. Kun IPS vertasi liikennettä signatuureja ja poikkeavuuksia vastaan, jotka määritettiin sääntökannassa, niin WAF tutkii enempi liikenteen käyttäytymistä ja logiikkaa. Se näkee mitä palvelimelta pyydetään ja mitä palvelin palauttaa. Tämän toimintamallin takia WAF pystyy suojelemaan web-sovellusta erilaisilta applikaatiotason tietoturvauhkilta, kuten SQL-injektioilta, Cross site scriptingiltä (XSS), yhteyden kaappaamiselta, parametri- ja URL-peukaloinnilta ja myös puskurin ylivuotovirheiltä. [21.]

WAF yleensä sijoitetaan tietoverkossa loogisesti web-palvelimen eteen. Tässä paikassa se kykenee tutkimaan liikennettä parhaiten ja pitämään yllä tietokantaa siitä, mitä liikennettä web-palvelimeen tulee ja mitä sieltä lähtee. WAF ei tässä tapauksessa näe kaikkea tietoverkon liikennettä, joten sitä käytetään yleensä vain tietyn loogisen osion suojaamiseeen. WAF:in kohdennettu käyttö on yksi sen parhaista puolista. IPS:t eivät

kykene tutkimaan liikennettä niin tarkasti kuin WAF, koska WAF:n loogisesti tarkkailema alue on huomattavasti pienempi. WAF:in kyky tarkkailla epäilyttävää ja odottamatonta liikenettä mahdollistaa täysin entuudeltaan tuntemattomien hyökkäyksien huomaamisen. [21.]

4 DAST (Dynamic Application Security Testing)

Web-sovelluksien tietoturva on nykyaikana erittäin kriittistä minkä tahansa yrityksen kannalta. Pilvipalveluiden lisääntyessä dramaattisin määrin on myös niiden turvallisuudesta herännyt paljon kysymyksiä. Kaikkien web-sovelluksien ehkäpä tärkein tietoturvaelementti on sovelluksien turvallinen koodi. Heikosti ohjelmoitu sovellus mahdollistaa eri haavoittuvuuksien käytön ja siten tietoturvahyökkäyksien tekemisen. Tämän takia sovelluksia testataan eri työkaluilla, ennekuin se otetaan yleisesti tuotantoon. Tämä ei kuitenkaan mahdollista täysin turvallista ohjelmaa, vaan aina saattaa löytyä jotain uusia haavoittuvuuksia esimerkiksi uusien ohjelmistopäivityksien myötä. Yleensä websovelluksien testaamiseen käytetään staattisia ja dynaamisia testejä. Näitä testausmenetelmiä kutsutaan DASTiksi ja SASTiksi. [22.]

4.1 Mitä DAST on?

DAST eli Dynamic Application Security Testing tarkoittaa sovelluksen koodin tietoturvan testaamista dynaamisin menetelmin. DAST pitää sisällään tietynlaiset työkalut, jotka testaavat sovellusta silloin kun se on käytössä. Kun esimerkiksi web-sovellusta testataan DASTilla niin DAST-järjestelmä yrittää tehdä sovellukseen muutoksia ja muuntaa sitä vaikkapa käyttäjän tavoin odottamattomilla menetelmillä, jotta tietoturvahaavoittuvuuksia saataisiin selville. Tämä on DASTin eroavaisuus SASTista. SAST eli Static Application Security Testing on joukko työkaluja, jolla itse lähdekoodia tarkkaillaan ja tutkitaan. Sovellus ei tällöin ole vielä käytössä, vaan sovelluksen lähdekoodi tutkitaan mahdollisien tietoturvahaavoittuvuuksien varalta. DAST:in tarkoituksena on siis suorittaa reaaliaikaista hyökkäystä sovellusta vastaan hyväksikäyttäen juuri käyttäjärajapintaa ja tällä tavalla tuottaa analyysia mahdollisista haavoittuvuuksista. [22.]

4.2 DASTin hyödyt ja haitat

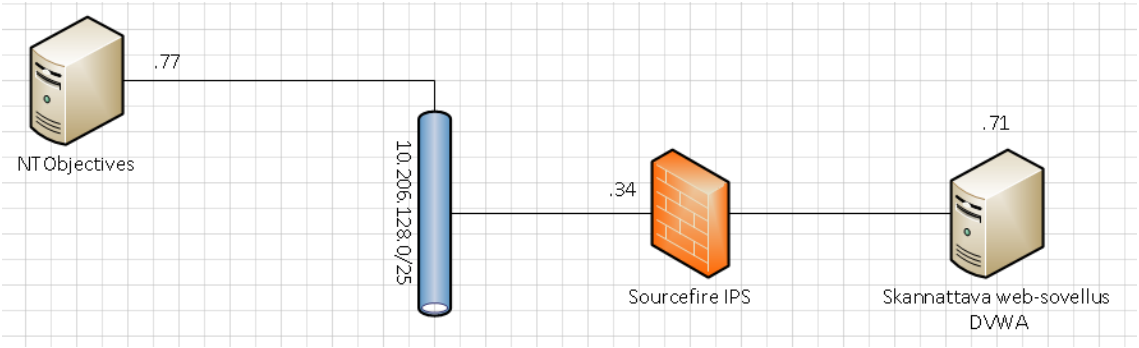
DAST on nykyään suuryrityksillä vakiotyökalu sovelluksen tietoturvan ja haavoittuvuuksien testaamisessa. Oikein käytettynä DAST pystyy auttamaan sovelluksen kehittäjiä monin eri tavoin, jotka muuten vaatisivat valtavia määriä ihmisresursseja. DAST:in tekemät skannaukset ja sovelluksen testaukset ovat hyvin laajoja ja kattavia. [23.]

DAST:issa piilee kuitenkin tiettyjä vaaroja, joita järjestelmänhallinnoijan täytyy varoa ja tärkeitä asioita, joita pitää tietää ja tehdä ennen sovelluksen dynaamista testausta. DAST:in käyttöä tuotannossa oleviin web-sovelluksiin ei suositella yleisesti, koska DAST voi aiheuttaa katkoksia palvelussa. Jos DAST:ia käytetään tällaisessa tapauksessa, on se parempi tehdä työaikojen ulkopuolella ja siitä täytyy informoida hyvin organisaation sisällä. Kaikesta testattavasta on parasta ottaa varmuuskopiot aina ennen testausta. Testattavan applikaation toimintaperiaate on myös tärkeä ymmärtää, mitä se tekee ja miten. Varsinkin tuotannossa olevien web-sovelluksien testaaminen, silloin kun se on pakollista, on hyvä rajata tiettyihin pienempiin osiin. Jokainen sovellus testataan siis erikseen. Tätä varten on erityisen tärkeää, että testaaja tietää DAST-järjestelmän konfiguroinnista mahdollisimman paljon. Tällöin dynaaminen testaus on älykkäästi kohdistettu ja tarkoituksellinen skannaus web-sovellusta vastaan. [23.]

5 Toteutus

Toteutusvaiheen tarkoitus on löytää DVWA (Damn Vulnerable Web Application) -palvelimelta mahdollisia web-sovellukseen liityviä haavoittuvuuksia, ja mikäli mahdollista, estää niiden hyväksikäyttäminen. Web-sovelluksen haavoittuvuudet ja niiden todentaminen tehdään DAST-järjestelmällä, joka on valittu tätä opinnäytetyötä varten. DAST-skannausjärjestelmä on NTObjectives–nimisen yrityksen tuote NTOSpider. NTOSpiderin valinta DAST-järjestelmäksi perustuu toteutuksen jälkimmäiseen vaiheeseen eli haavoittuvuuksien hyväksikäytön estämiseen. NTObjectives on julkaissut ohjelman nimeltä NTODefendin, jonka avulla voi luoda sääntöjä erilaisiin tuettuihin IPS/WAF–järjestelmiin. NTODefender pystyy tulkitsemaan NTOSpiderin luomaa listaa haavoittuvuuksista ja kääntämään ne tuettuun IPS/WAF–järjestelmään säännöiksi. Toteutusvaiheessa käytetään Sourcefire–yrityksen laitetta, joka toimii IPS-tilassa.

Toteutusvaiheessa etsitään siis mahdollisuuksia vaikuttaa web-sovelluksen tietoturvaan ja sen parantamiseen etsien mahdolliset haavoittuvuudet ja paikaten ne käyttäen DAST-järjestelmän ja IPS:n välistä haavoittuvuuskannan muuntamista sääntökannaksi. Tämä on haluttua silloin, kun mahdollisesti kyseiseen sovellukseen ei pystytä tekemään heti muutoksia sen kannattavuuden tai hankaluuden vuoksi. Tämä web-sovelluksen haavoittuvuuksien estäminen käyttäen IPS:ää tuo kokonaan erilaisen näkökulman sovelluskohtaiseen tietoturvaan ja nopean vasteajan toiminnalle IPS:n tapahtumahälytyksien avulla, varsinkin jos kyseessä on kriittinen sovellusjärjestelmä.



Kuva 7. Kuva loogisesta IP-verkkotopologiasta. DVWA web-sovellus sijaitsee IP-osoitteessa 10.206.128.71.

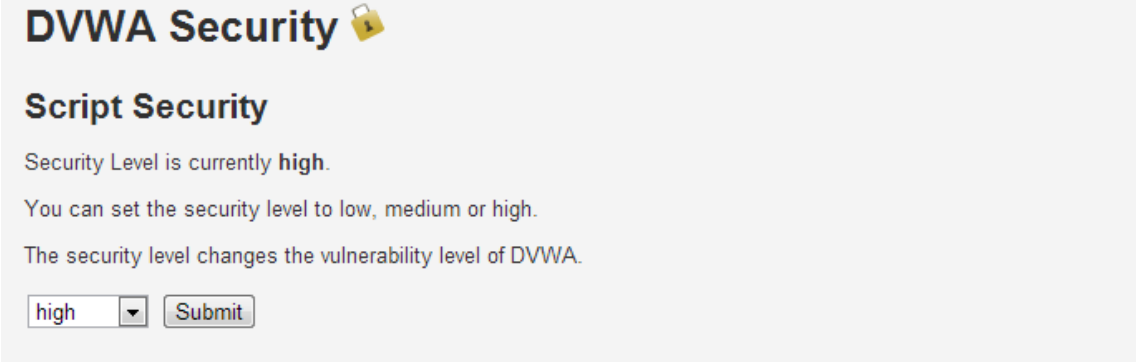
5.1 Liferay 6

Liferay 6 on web-sovellus, joka toimii Apache web-alustan päällä testiympäristössä. Käyttöjärjestelmänä tällä palvelimella on Centos 6.3. Cygate Oy:lla on tarkoituksena mahdollisesti ottaa käyttöön Liferay 6 tiettyjä web-palveluita varten. Tämän takia web-sovelluksen tietoturva on erityisen tärkeässä roolissa sen käyttöönotossa. Liferay 6 on suojattu OWASP 2010 -haavoittuvuuksia vastaan tuoteselostuksen mukaan ja tämä voidaan varmistaa DAST-skannauksilla web-sovellusta kohtaan mahdollisesti myöhemmin. Tässä opinnäytetyössä yhtenä tarkoituksena on varmistaa DAST-järjestelmän luotettava toiminta, jotta sitä voidaan mahdollisesti hyödyntää myöhemmin kartoittamaan muiden web-sovelluksien, kuten Liferay 6:n, tietoturvatasot ja mahdolliset ongelmat.

5.2 DVWA eli Damn Vulnerable Web Application

DAST-skannauksien kohteeksi valittiin web-sovellus DVWA, joka toimii myös Apache web-alustan päällä. DVWA on nimensä mukaisesti hyvin paljon tietoturva-aukkoja sisäl-

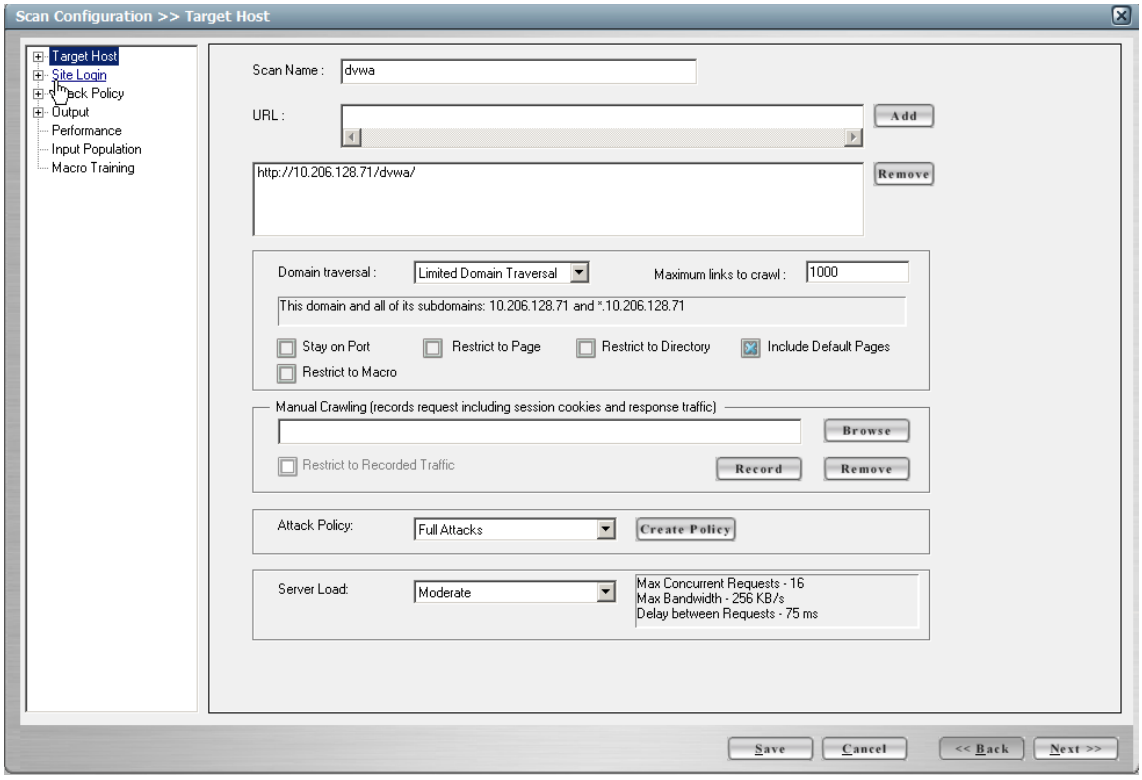
tävä web-sovellus. Tärkein syy DVWA:n valintaan oli DAST-skannauksien tulosten yksinkertaisempi validoiminen. DVWA:n tietoturva-asetuksia on myös hyvin helppo vaihtaa ja DVWA sisältää oman PHP IDS -järjestelmän. Tätä järjestelmää ei kuitenkaan haluttu enabloida tätä työtä varten.



Kuva 8. Kuvankaappaus palvelimen tietoturva-asetuksista DAST-skannausta varten. Tietoturvatasoksi valittiin korkea turhien haavoittuvuuksien poissulkemisen takia.

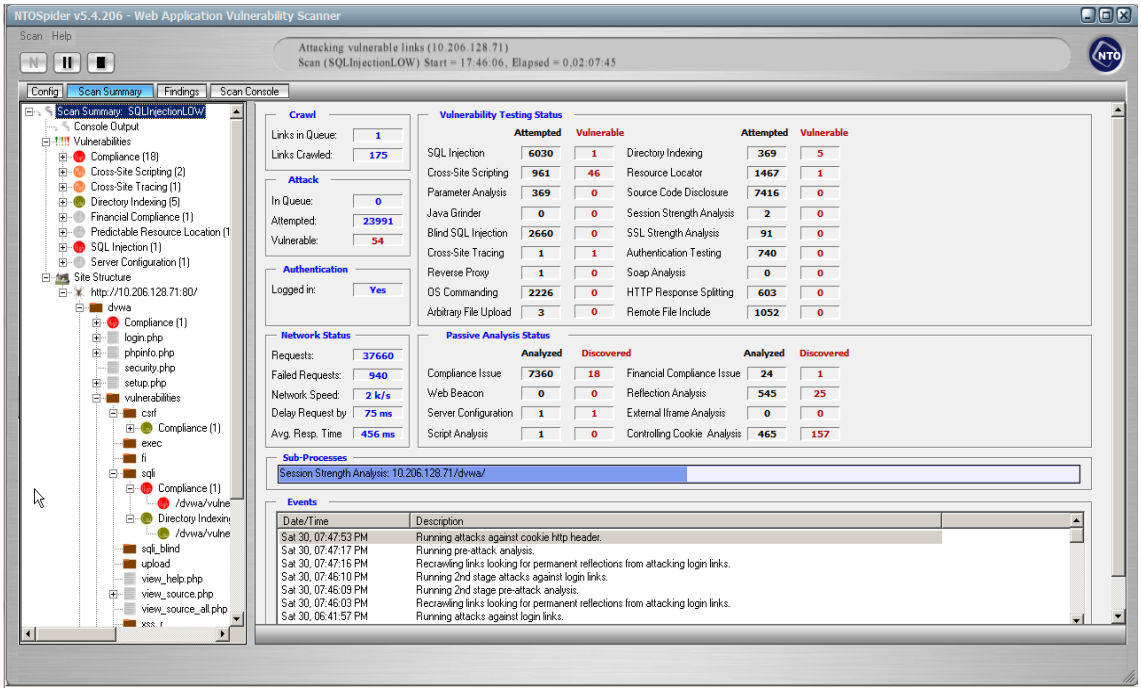
5.3 NTOSpider ja haavoittuvuuksien etsiminen

NTOSpiderin lisenssin saamisen jälkeen ohjelmisto asennettiin Windows server 2008 R2 käyttöjärjestelmällä toimivalle virtuaalipalvelimelle. Tästä palvelimesta skannattiin kohdepalvelimet. Skannauksien suorittaminen on suhteellisen yksinkertaista. Perustason skannauksia pystyy suorittamaan hyvinkin nopeasti ohjelman käynnistyttyä.



Kuva 9. Skannauksen ensimmäinen sivu, josta pystyy määrittämään kaiken tarvittavan ensimmäistä skannausta varten. Hyökkäyspolitiikkana käytettiin kaikki haavoittuvuudet tarkastavaa Full Attacks -politiikkaa. Palvelimeen kohdistuvan skannauksen aiheuttama kuorma määritettiin keskisuureksi.

Skannauksen voi aloittaa heti, kun tarpeelliset asetukset on määritetty. Hyökkäyspolitiikan kannattaa vastata palvelimen toimintaa, jotta turhilta viiveiltä vältytään. Tämän jälkeen skannauksen tuloksia voi seurata eri välilehdiltä, jotka näkyvät NTOSpiderissä. Välilehdistä löytyy skannauksen yleinen tila, löydökset ja konsoli-pohjainen näkymä, jossa näkyy DAST-järjestelmän tekemät hyökkäykset.

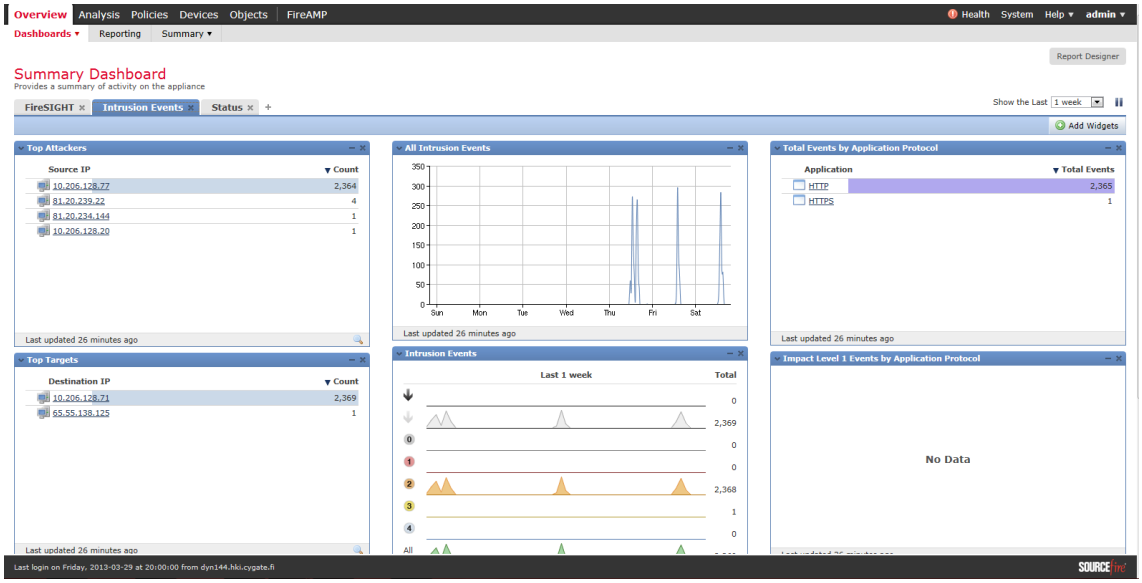


Kuva 10. Skannauksen yleinen tila. Tämä välilehti antaa kattavan yleiskuvan skannauksen tilasta ja mahdollisista löydöksistä. Vasemmalla näkyy web-sovelluksen sivut juurirakenteena.

Skannauksen jälkeen NTOSpider luo kattavan HTML-raportin. Raportti pitää sisällään kaikki löydökset web-sovelluksesta. Raportin voi jakaa useaan eri osaan ja sitä voi katsella eri osa-alueiden kannalta. Tämä tarkoittaa sitä, että mahdollisia haavoittuvuuksia voi katsoa tietokannan hallinnoijan kannalta tai sitten itse web-sovelluksen ohjelmoijan näkökulmasta. Raportti antaa myös kustannusarvion haavoittuvuuksien korjaamisesta. NTOSpider luo samalla XML-tiedoston nimeltä VulnerabilitiesSummary. Tämä tiedosto pitää sisällään tiedot haavoittuvuuksista, joiden hyödyntämistä vastaan voidaan luodaan säännöt IPS/WAF-järjestelmään NTODefendillä.

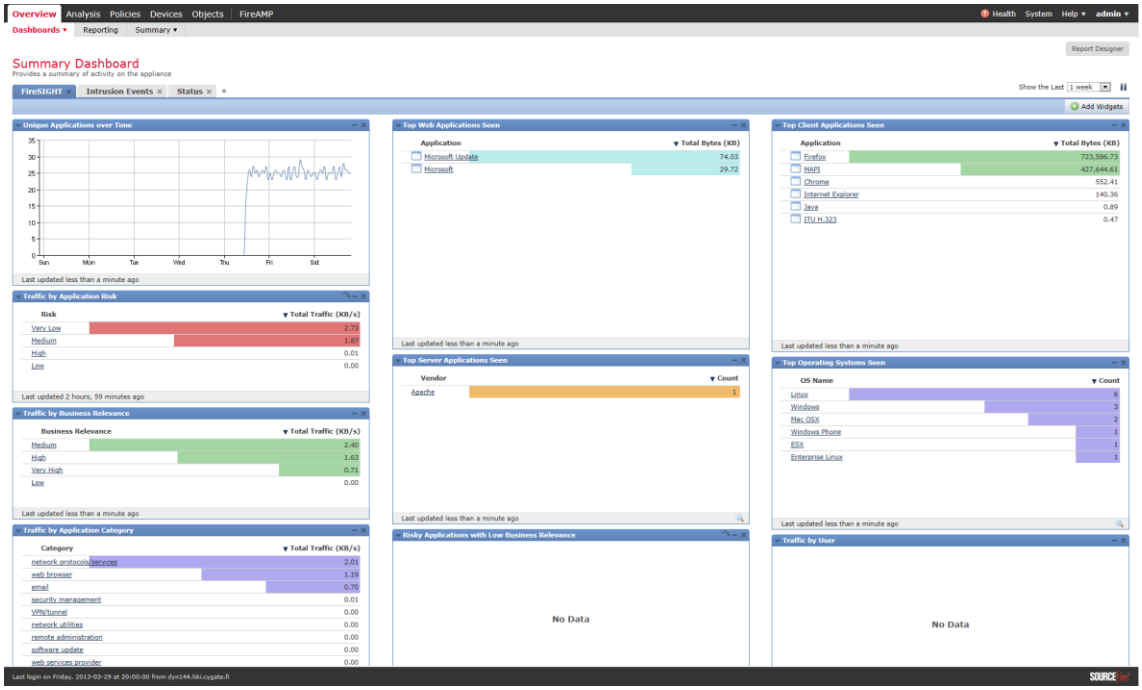
5.4 Sourcefire IPS

Toteutusvaiheeseen valittiin IPS-järjestelmäksi Sourcefire, joka pohjautuu vapaan lähdekoodin Snort-ohjelmistoon. Sourcefire on tällä hetkellä markkinajohtaja, ja se tekee yhteistyötä NTObjectivesin kanssa. NTODefend mainitsee yhdeksi tukemakseen IPS-järjestelmäksi juuri Sourcefiren. IPS-järjestelmään kuuluu kaksi osaa tässä opinnäytetyössä, jotka ovat hallintapalvelin eli Sourcefire Defence Center ja IPS-sensori.



Kuva 11. Sourcefire Defence Centerin etusivu. Tällä sivulla olevat graafiset taulukot voidaan itse valita suuresta määrästä eri taulukoita. Vaihtoehtoina on esimerkiksi tapahtumahälytykset lukumäärinä, eniten hyökätyt kohteet ja eniten hyökänneet lähteet.

Sourcefiressä käyttää FireSIGHT-nimistä teknologiaa, joka monitoroi tietoverkossa kulkevaa dataa ja etsii siitä tiettyjä asioita, kuten esimerkiksi lähde- ja kohdeosoitteiden käyttöjärjestelmiä, eniten käytettyjä sovelluksia ja käytettyjä selaimia. Näiden pohjalta FireSIGHT kykenee suosittelemaan tiettyjä sääntöjä otettavaksi käyttöön. Hyvänä esimerkkinä on Apache web-alusta, jonka pohjalta on otettu käyttöön Apachen haavoittuvuuksia tarkkailevat säännöt.



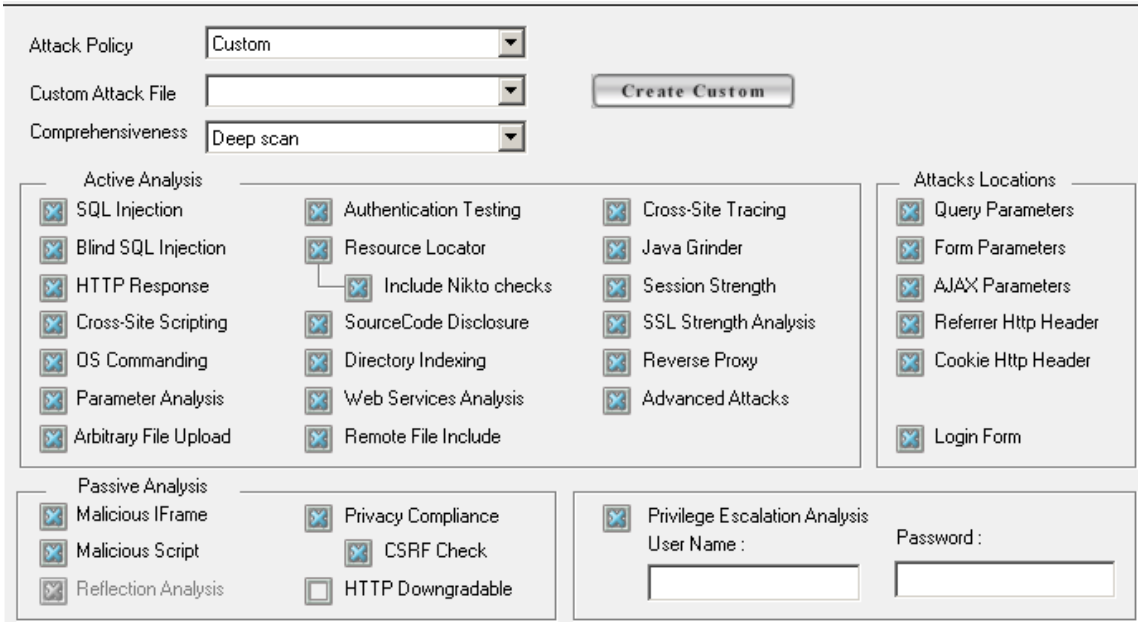
Kuva 12. FireSIGHT-näkymä Sourcefire Defence Centerissä. Kuten muilla välilehdillä, myös tällä sivulla voi määrittää ne graafiset taulukot, joita halutaan seurata.

Sourcefiren IPS-sääntökannassa määritetään sääntöpolitiikka, jolle annetaan nimi ja peruspolitiikka. Peruspolitiikassa määritetään laitevalmistajalta tullut pohjasääntökanta, valintoina ovat Balanced Security And Connectivity, Connectivity Over Security ja Security Over Connectivity. IPS:n sääntöpolitiikassa on kolme eri tasoa, joissa ylemmän tason säännöt korvaavat aina alemman tason säännöt. Alimpana tasona on siis pohjasääntökanta, seuraavaksi keskitasolla on FireSIGHT-järjestelmän ehdottamat säännöt ja korkeimmalla tasolla on käyttäjän itse tekemät säännöt ja myös DAST-järjestelmästä luodut säännöt. Tarkoituksena tällä on se, että käyttäjän säännöt aina menevät muiden sääntöjen päälle ja käyttäjän tekemät muutokset tulevat voimaan.

6 Tulokset ja johtopäätökset

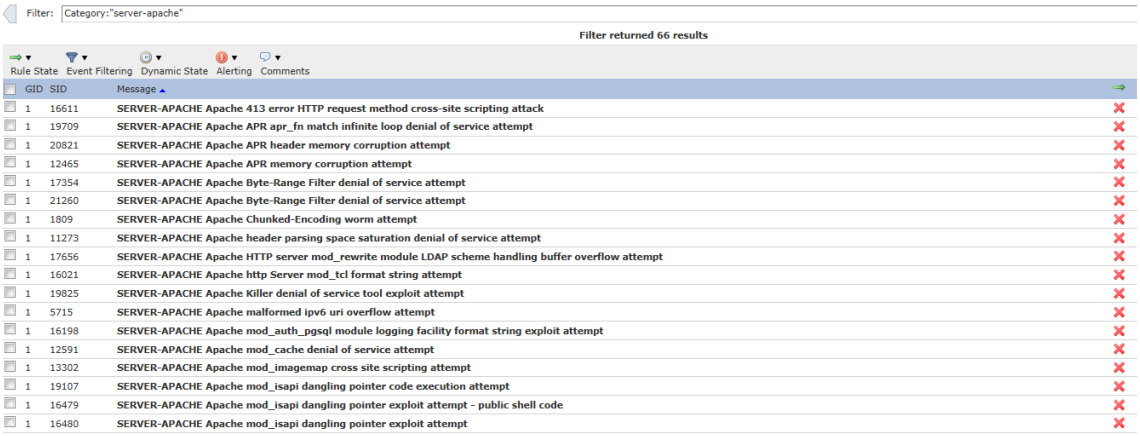
6.1 Ensimmäinen skannaus

Ensimmäisessä skannauksessa DVWA-sovelluksen tietoturva-asetuksia pidettiin alhaisena, jotta myös yksinkertaisimmat haavoittuvuudet saataisiin esille ja tuloksista mahdollisimman kattavat. NTOSpiderissa skannaukseksi valittiin täysi hyökkäys, jotta web-sovelluksen mahdolliset haavoittuvuudet saataisiin parhaiten selville. Tämä skannausmenetelmä kertoo paljon web-sovelluksesta ja sen muista ongelmista – myös niistä, joita vastaan IPS-järjestelmällä ei voi suojautua. Yleisesti tällaisia asioita on palvelimen palauttamat vastaukset tiettyihin pyyntöihin ja esimerkiksi mahdolliset kyselyt, joilla voidaan saada selville palvelimen web-alusta (Tässä tapauksessa Apache).



Kuva 13. Hyökkäyspolitiikka NTOSpiderissa. Kuvasta näkee, että kaikki mahdolliset hyökkäysmenetelmät ja haavoittuvuudet halutaan saada selville. NTOSpiderin haluttiin hyödyntävän kaikki mahdolliset lokaatiot, josta voi hyväksikäyttää palvelimen haavoittuvuuuksia.

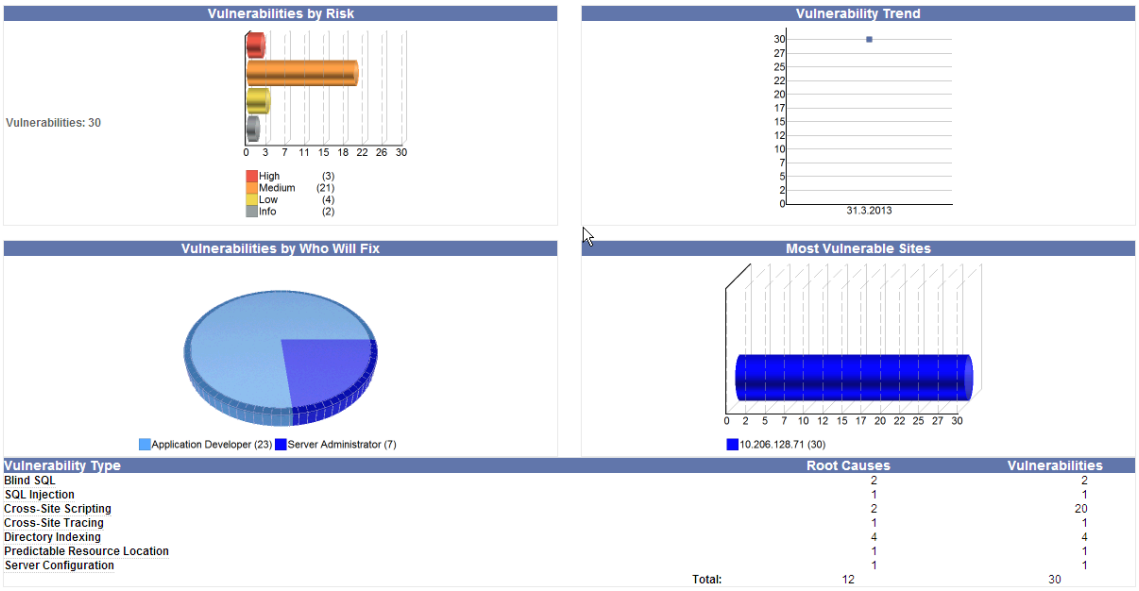
IPS-järjestelmä asetettiin IDS-tilaan, jolloin se huomaa mahdolliset hyökkäykset, mutta ei kuitenkaan estä niitä. Tällä tavalla varmistettiin skannauksen onnistuminen. Samalla Sourcefiren lokitusta seuraamalla pystyttiin nähdä DAST-järjestelmän suorittamat hyökkäykset ja niiden määrä, mikäli Sourcefire kyseiset hyökkäykset huomasi. Jotta Sourcefire huomaisi mahdolliset hyökkäykset, on se määritettävä sääntöpolitiikassa ottamaan kantaa kyseisiin hyökkäyksiin.



Kuva 14. Kuvankaappaus pienestä osasta sääntöjä, jotka on otettu käyttöön sääntöpolitiikassa liittyen Apache-web-alustan haavoittuvuuksiin. Punainen rasti oikealla tarkoittaa liikenteen estämistä ja siitä hälytyksen luomista. Liikennettä ei kuitenkaan ensimmäisessä skannauksessa estetä, koska itse sääntöpolitiikassa on määritetty IPS-toiminnallisuus pois päältä.

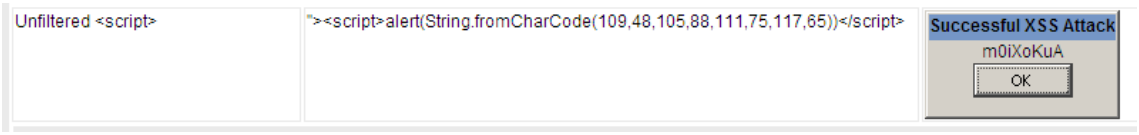
Tärkeimpänä asiana ensimmäisessä skannauksessa oli siis nimenomaan web-sovelluksen haavoittuvuuksien löytäminen ja niistä raportin luominen. Valmiin skannauksen jälkeen NTOSpider luo HTML-pohjaisen raportin automaattisesti ja sen voi määrittää luomaan myös PDF-tiedoston haavoittuvuuksista.

Ensimmäisessä skannauksessa löytyi 30 haavoittuvuutta, joista kolme olivat korkean prioriteetin haavoittuvuuksia ja 21 haavoittuvuutta olivat keskisuuren prioriteetin haavoittuvuuksia. Loput haavoittuvuudet jakautuivat kuvion 15 mukaan.



Kuva 15. Kuva NTOSpiderin luoman raportin etusivusta. Kuvasta käy selvästi ilmi, että palvelimella on huomattava määrä haavoittuvuuksia. Haavoittuvuuksien eri lajittelumetodit ovat kattavat, esimerkiksi vastuualueittain.

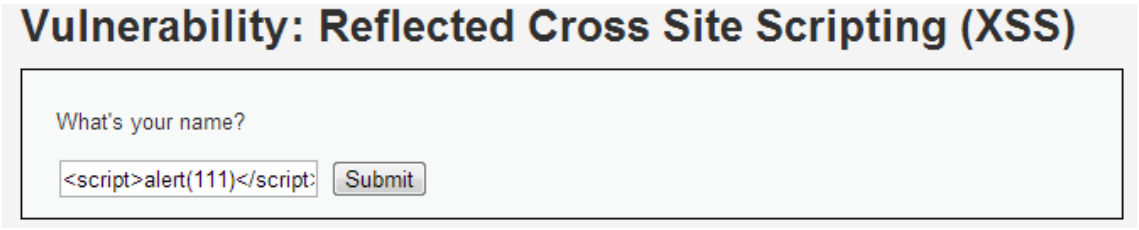
Ensimmäisen skannauksen valmistumisen jälkeen tulokset käytiin läpi ja osa tuloksista validoitiin. Validoimisen tarkoituksena on tulosten oikeellisuuden tarkastaminen. Tämä on erityisen tärkeää, koska tällä tavalla voidaan suorittaa tarkempi riskikartoitus web-sovelluksen tietoturvasta. Ensimmäisen skannauksen validoimiseksi valittiin XSS-haavoittuvuuden hyväksikäyttö. Tässä vaiheessa nousi esiin NTOSpiderin erittäin hyvä toiminto, haavoittuvuuksien validoiminen. NTOSpider näyttää raportissa suoraan, minkä takia se on merkinnyt haavoittuvuuden olemassa olevaksi ja näyttää komennot, joita se on lähettänyt web-sovellukseen.



Kuva 16. NTOSpider raportoi XSS-haavoittuvuudesta, jossa se kykenee suorittamaan JavaScript-komentoja, koska lomakkeessa ei suodateta tiettyjä erikoismerkkejä.

Tärkeää on myös havainnoida, että NTOSpider siis suorittaa XSS-hyökkäyksen, eli se validoi sen jo ennen kuin se ilmoittaa haavoittuvuudesta. Tämän takia on hyvin tärkeää, että skannattavalle web-sovellukselle on tehty tarvittavat toimenpiteet ja mahdolliset riskit on otettu huomioon, kuten on mainittu kappaleessa 4.2.

XSS-haavoittuvuus validoitiin hyvin yksinkertaisella menetelmällä, jossa web-sovellukseen yritettiin lähettää JavaScript-komento. Validointi onnistui ja huomattiin, että web-sovellus antaa käyttäjän suorittaa JavaScript-komentoja lomakkeessa.



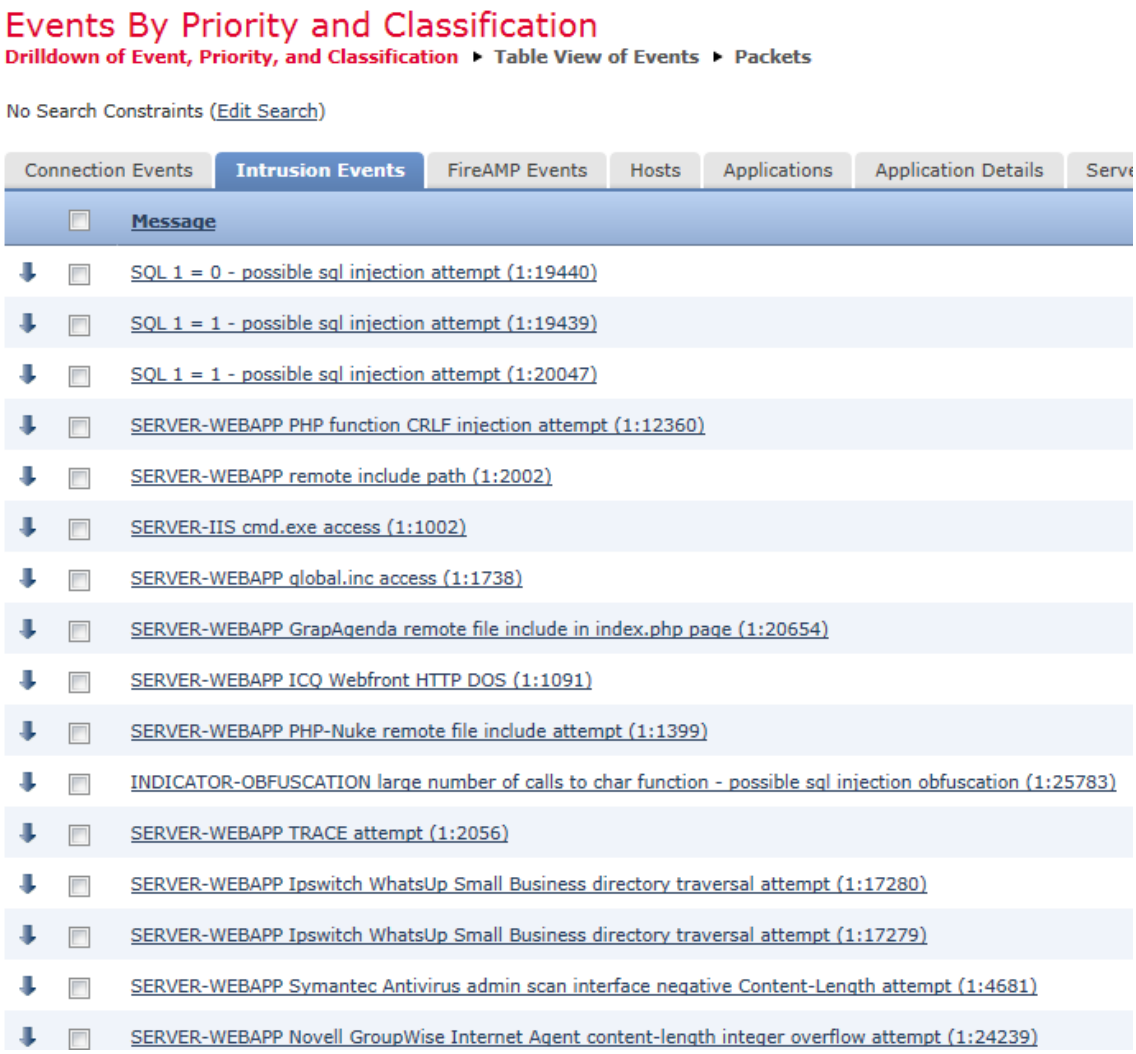
Kuva 17. Web-sovelluksessa olevaan lomakkeeseen sijoitettiin JavaScript-komento <script>alert(111)</script>

Tämän jälkeen lähdekoodia katsomalla, voitiin varmistua, että XSS-haavoittuvuuden hyväksikäyttäminen onnistui ja komento saatiin ajettua.

```
<div class="vulnerable_code_area">

        <form name="XSS" action="#" method="GET">
                <p>What's your name?</p>
                <input type="text" name="name">
                <input type="submit" value="Submit">
        </form>

        <pre>Hello <script>alert(111)</script></pre>

</div>
```

Kuva 18. Kuvassa hyvin yksinkertaistettu XSS-haavoittuvuuden hyväksikäyttö. Lomakkeessa sallitaan erikoismerkit, joita ei siis suodateta. Tästä aiheutuu esimerkiksi mahdollisuus ajaa erilaisia komentoja tietokantaa vastaan.

Sourcefiren lokituksen avulla voitiin todentaa myös, että NTOSpider todellakin hyökkää web-sovellusta vastaan. Lokituksessa nähdään myös, kuinka monta kertaa kyseinen hyökkäys on tapahtunut ja hyökkäystä on mahdollista tutkia pakettitasolla asti.



Kuva 19. Sourcefiren lokitusta hälytyksistä. Tässä näkymässä nähdään mahdollisesti tapahtuneet hyökkäykset tietyltä aikajanalta ja niiden lukumäärät.

Packet Information

FRAME 1 (Expand All)

- **Frame 1** (619 bytes captured)
- **Ethernet II** (Src: 00:0c:29:eb:d6:60, Dst: 00:0c:29:d0:0d:fb)
- **Internet Protocol** (Src: 10.206.128.77, Dst: 10.206.128.71)
- **Transmission Control Protocol** (Src Port: 56091 (56091), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 565)
- **Packet Text**
- **Packet Bytes**

```
0000  00 0c 29 d0 0d fb 00 0c 29 eb d6 60 08 00 45 00  ..).....)..`..E.
0010  02 5d 0e 21 40 00 80 06 d4 49 0a ce 80 4d 0a ce  .].!@....I...M..
0020  80 47 db 1b 00 50 ec 38 b9 47 c8 7b 3e f3 50 18  .G...P.8.G.{>.P.
0030  01 00 e0 7f 00 00 47 45 54 20 2f 64 76 77 61 2f  ......GET /dvwa/
0040  76 75 6c 6e 65 72 61 62 69 6c 69 74 69 65 73 2f  vulnerabilities/
0050  62 72 75 74 65 2f 3f 75 73 65 72 6e 61 6d 65 3d  brute/?username=
0060  61 64 6d 69 6e 27 2b 4f 52 2b 31 3d 30 2b 23 26  admin'+OR+1=0+#&
0070  70 61 73 73 77 6f 72 64 3d 6d 32 32 4f 33 58 67  password=m22O3Xg
0080  55 26 4c 6f 67 69 6e 3d 4c 6f 67 69 6e 20 48 54  U&Login=Login HT
0090  54 50 2f 31 2e 31 0d 0a 55 73 65 72 2d 41 67 65  TP/1.1..User-Age
00a0  6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20  nt: Mozilla/5.0
00b0  28 57 69 6e 64 6f 77 73 3b 20 55 3b 20 57 69 6e  (Windows; U; Win
00c0  64 6f 77 73 20 4e 54 20 35 2e 32 3b 20 65 6e 2d  dows NT 5.2; en-
00d0  55 53 3b 20 72 76 3a 31 2e 39 2e 31 2e 35 29 20  US; rv:1.9.1.5)
00e0  47 65 63 6b 6f 2f 32 30 30 39 31 31 30 32 20 46  Gecko/20091102 F
00f0  69 72 65 66 6f 78 2f 33 2e 35 2e 35 0d 0a 48 6f  irefox/3.5.5..Ho
0100  73 74 3a 20 31 30 2e 32 30 36 2e 31 32 38 2e 37  st: 10.206.128.7
0110  31 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61  1..Accept-Langua
0120  67 65 3a 20 65 6e 2d 75 73 0d 0a 43 6f 6e 74 65  ge: en-us..Conte
0130  6e 74 2d 54 79 70 65 3a 20 61 70 70 6c 69 63 61  nt-Type: applica
0140  74 69 6f 6e 2f 78 2d 77 77 77 2d 66 6f 72 6d 2d  tion/x-www-form-
0150  75 72 6c 65 6e 63 6f 64 65 64 0d 0a 41 63 63 65  urlencoded..Acce
0160  70 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c 61 70  pt: text/html,ap
0170  70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d 6c 2b  plication/xhtml+
0180  78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f  xml,application/
0190  78 6d 6c 3b 71 3d 30 2e 39 2c 2a 2f 2a 3b 71 3d  xml;q=0.9,*/*;q=
01a0  30 2e 38 0d 0a 43 61 63 68 65 2d 43 6f 6e 74 72  0.8..Cache-Contr
01b0  6f 6c 3a 20 6e 6f 2d 63 61 63 68 65 0d 0a 52 65  ol: no-cache..Re
01c0  66 65 72 65 72 3a 20 68 74 74 70 3a 2f 2f 31 30  ferer: http://10
01d0  2e 32 30 36 2e 31 32 38 2e 37 31 2f 64 76 77 61  .206.128.71/dvwa
01e0  2f 76 75 6c 6e 65 72 61 62 69 6c 69 74 69 65 73  /vulnerabilities
01f0  2f 62 72 75 74 65 2f 0d 0a 43 6f 6f 6b 69 65 3a  /brute/..Cookie:
0200  20 50 48 50 53 45 53 53 49 44 3d 68 68 39 65 66  PHPSESSID=hh9ef
0210  39 30 30 67 6f 6f 36 66 63 72 6e 64 6d 6c 32 64  900goo6fcrndml2d
0220  69 76 70 65 36 3b 20 73 65 63 75 72 69 74 79 3d  ivpe6; security=
0230  68 69 67 68 0d 0a 41 63 63 65 70 74 2d 45 6e 63  high..Accept-Enc
0240  6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66  oding: gzip, def
0250  6c 61 74 65 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e  late..Connection
0260  3a 20 43 6c 6f 73 65 0d 0a 0d 0a                 : Close....
```

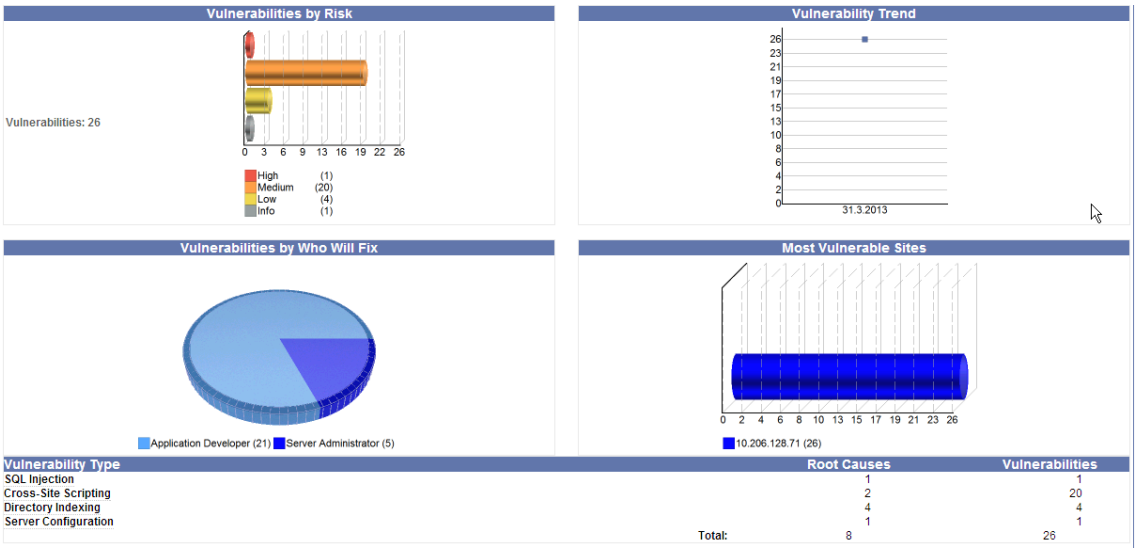
Kuva 20. Hyökkäyksen tarkastelu pakettitasolla. Kuvasta näkee selvästi, kuinka NTOSpider on yrittänyt SQL-injektiota web-sovellukseen.

Ensimmäisen skannauksen tulokset olivat tyydyttäviä, ja ne vahvistivat epäilykset siitä, että DVWA web-sovelluksen tietoturva-asetuksien ollessa alhaiset, haavoittuvuuksia on useita. Tulokset antoivat myös kattavan pohjan seuraaville skannauksille. NTOSpider yrittää esimerkiksi SQL-injektioita ja XSS-hyökkäyksiä, joten näiden haavoittuvuuksien korjaamisella ja niiden hyväksikäytön estämisellä on suora vaikutus skannauksessa löytyviin haavoittuvuuksiin, niiden vähenemisiin ja pyyntöjen epäonnistumisiin.

6.2 Toinen skannaus

Toisessa skannauksessa tarkoituksena oli nähdä IPS:n oma kyky estää mahdolliset haavoittuvuudet ja niiden hyväksikäyttäminen. Koska NTOSpider suorittaa reaaliaikaisia hyökkäyksiä haavoittuvuuksien löytämiseen, Sourcefire pystyy estämään näitä, jos sen sääntöpolitiikassa on niin määritetty. Sääntöpolitiikka pidettiin toisessa skannauksessa samana kuin ensimmäisessä skannauksessa, mutta tällä kertaa IPS-toiminallisuus oli laitettu päälle.

Toisen skannauksen valmistuttua NTOSpiderin luoma raportti käytiin läpi. Pieni osa haavoittuvuuksista oli hävinnyt, mikä on merkki siitä, että Sourcefire kykenee omatoimisesti estämään haavoittuvuuksia. Kuitenkin suurin osa haavoittuvuuksista edelleenkin näkyi raportissa.



Kuva 21. Toisen skannauksen tulokset. IPS on estänyt muutamia haavoittuvuuksia, mutta suurin osa haavoittuvuuksia silti löytyi.

Priority	Inline Result	Source IP	Destination IP	Classification
medium	↓	10.206.128.77	10.206.128.71	Attempted Information Leak
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Attempted User Privilege Gain
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
medium	↓	10.206.128.77	10.206.128.71	Attempted Information Leak
medium	↓	10.206.128.77	10.206.128.71	Access to a Potentially Vulnerable Web Application
medium	↓	10.206.128.77	10.206.128.71	Attempted Information Leak
high	↓	10.206.128.77	10.206.128.71	Attempted User Privilege Gain
medium	↓	10.206.128.77	10.206.128.71	Attempted Information Leak
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Attempted User Privilege Gain
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
medium	↓	10.206.128.77	10.206.128.71	Attempted Information Leak
high	↓	10.206.128.77	10.206.128.71	Web Application Attack
high	↓	10.206.128.77	10.206.128.71	Attempted User Privilege Gain

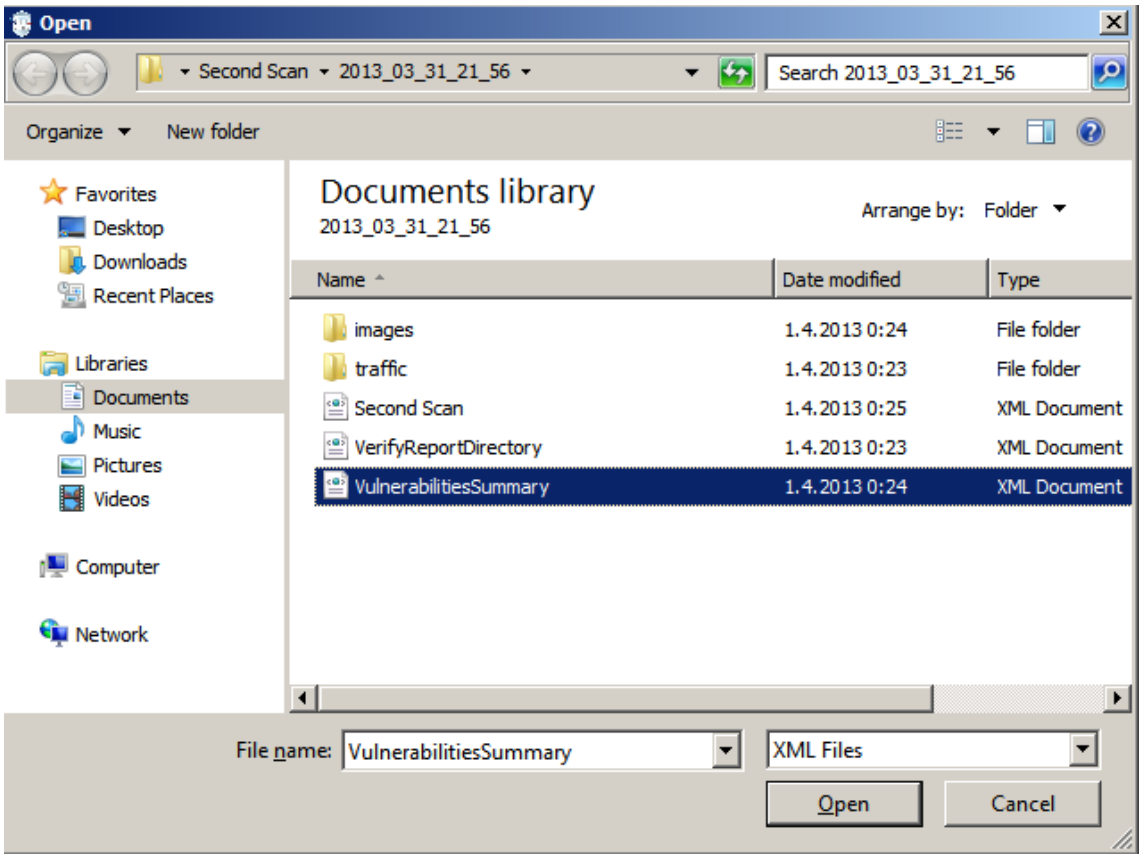
Kuva 22. Lokitusnäkymä Sourcefirestä. Musta nuoli, joka osoittaa alaspäin, on merkki siitä, että liikenne estetään. Sourcefire tunnistaa osan NTOSpiderin suorittamista hyökkäyksistä ja pystyy määrittämään niiden tarkoituksen osittain.

Haavoittuvuuksien kokonaismäärä oli toisen skannauksen jälkeen 26. IPS ei siis pystynyt täysin estämään haavoittuvuuksia, vaikka se osan pystyikin estämään. Tämä johtuu suurimmaksi osaksi siitä, että IPS toimii tiettyjen sääntöjen pohjalta, jotka tutkivat liikennettä. Koska säännöt pitäisi määritellä manuaalisesti ja tämä vaatii IPS:n hallinnoijalta erittäin kattavaa tietämystä itse web-sovelluksesta, ei se ole yleensä mahdollista. Sourcefiressä oleva FireSIGHT-järjestelmä ei ole web-sovelluksen testiväline, vaan

enemmänkin verkossa yleisimpien liikennöitsijöiden, sovellusten ja selainten monitorointijärjestelmä. Se ei siis pysty antamaan täydellistä kuvaa niistä säännöistä, jotka web-sovelluksen tietoturvan vuoksi pitäisi ottaa käyttöön.

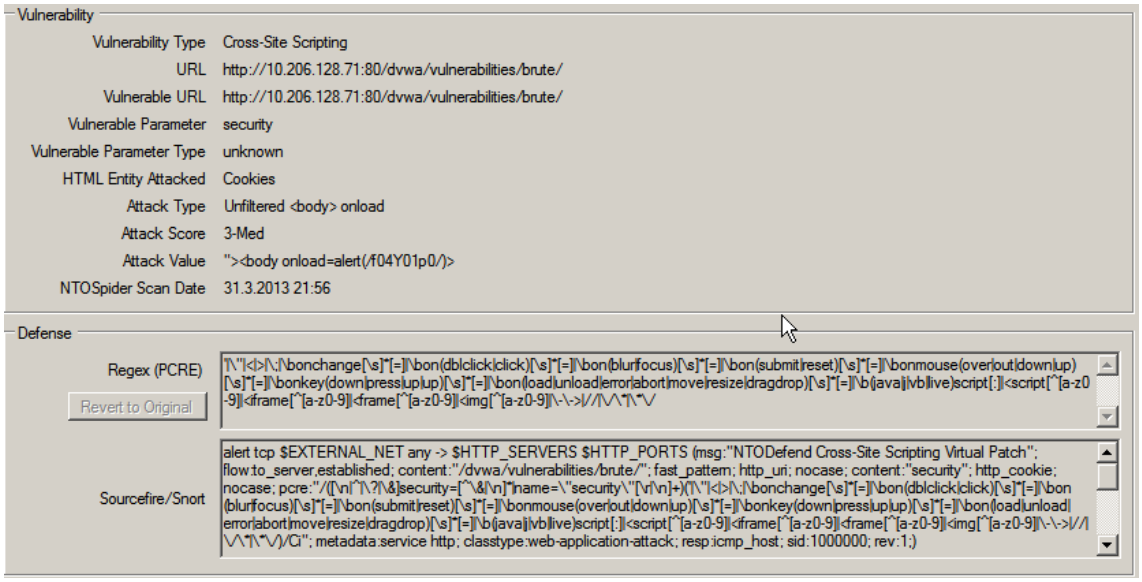
6.3 Kolmas skannaus

Toisesta skannauksesta luotu haavoittuvuusraportti luo myös XML-tiedoston VulnerabilitiesSummary.xml, jonka voi aukaista NTODefend-ohjelmalla. NTODefend kääntää halutun haavoittuvuuslistan säännöiksi valitulle IPS/WAF-järjestelmälle.



Kuva 23. Haavoittuvuuslista löytyy NTOSpiderin erikseen määräämästä tiedostosijainnista toisen skannauksen kansion alta.

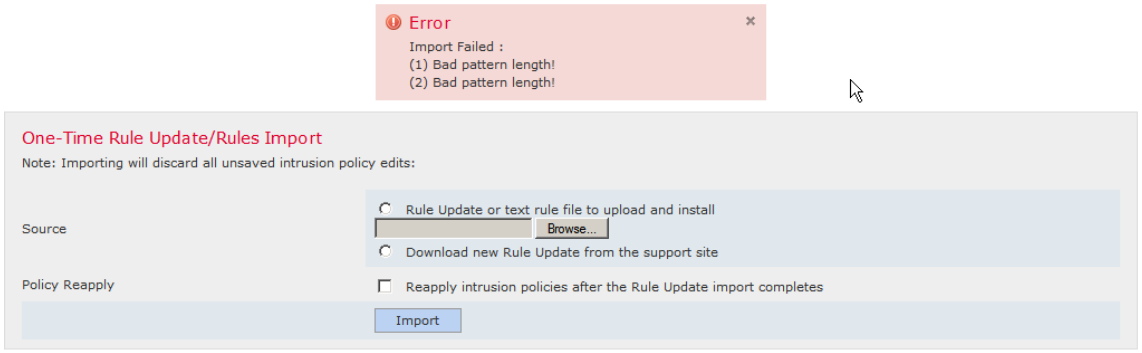
NTODefendissä valittiin IPS-järjestelmäksi Sourcefire/Snort, jonka säännöiksi siis NTODefend löydetyt haavoittuvuudet muuntaa.



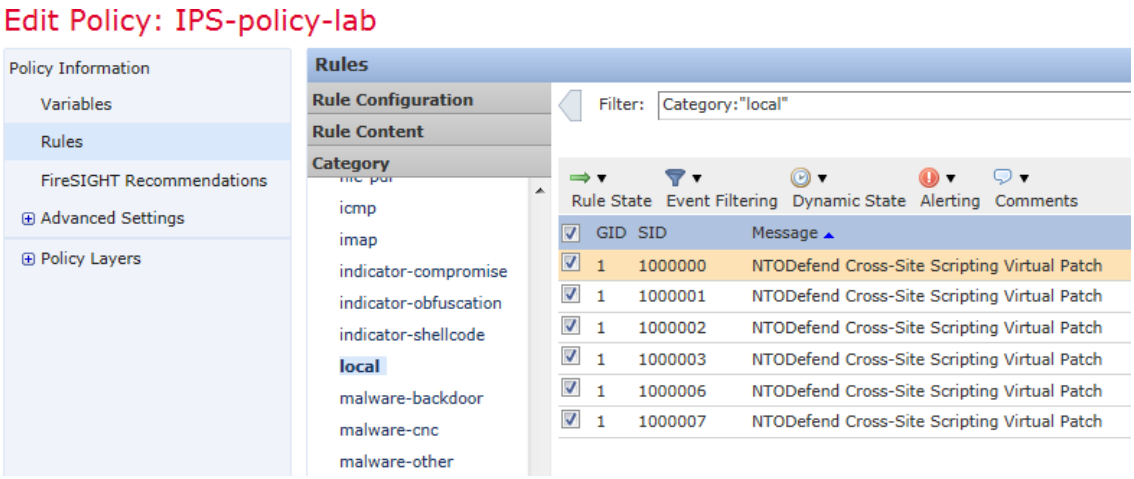
Kuva 24. Kuvankaappaus NTODefendin etusivulta haavoittuvuuslistan lataamisen jälkeen. Kuvasta näkee, miten NTODefend luo säännön Sourcefireä varten kuvan alalaidassa

NTODefend luo tiedoston, joka siirretään Sourcefireen. Tiedosto pitää sisällään syntaksiltaan Snort-sääntöjä. Säännöt voi siirtää myös tekstinä avaamalla NTODefendin luoman tiedoston esimerkiksi notepadilla ja siitä kopioimalla tekstitiedoston Sourcefiren Rule Editoriin.

Kaikkia DAST-sääntöjä ei kuitenkaan valitettavasti saanut suoraan siirrettyä IPS:ään, koska Sourcefire ei hyväksynyt tiettyjä sääntöjä.



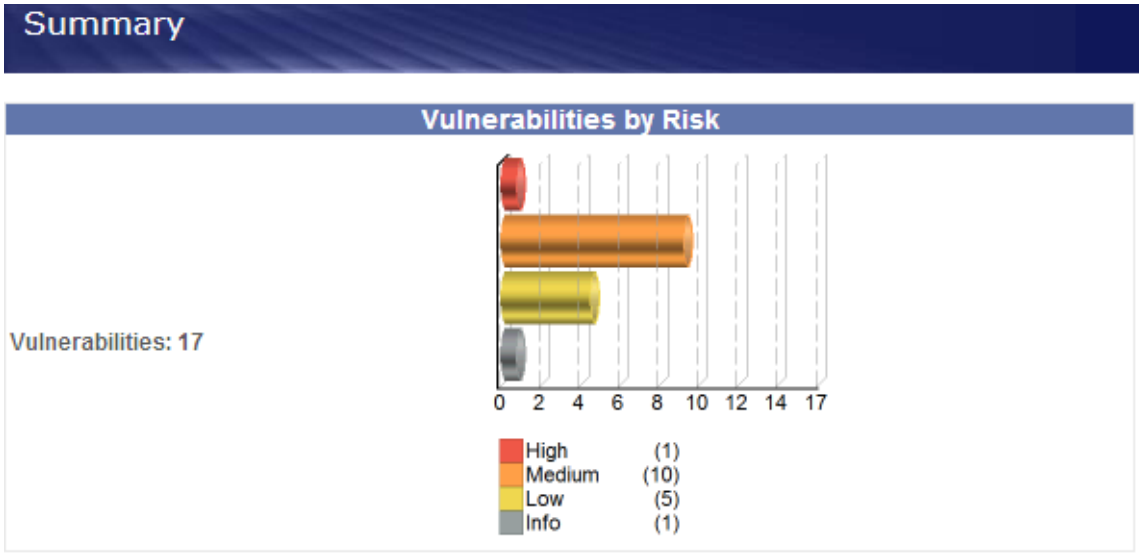
Kuva 25. Kuva Sourcefiren antamasta herjasta. Osassa NTODefendin luomissa säännöissä on syntaksivirheitä ja tämän vuoksi niitä ei saatu siirrettyä IPS:ään. Suurimman osan säännöistä kuitenkin sai siirrettyä ilman ongelmia.



Kuva 26. Sääntöjen lisäämisen jälkeen itse sääntöpolitiikassa pitää käydä ottamassa säännöt käyttöön. Sourcefireen NTODefendistä tuodut säännöt ilmestyvät local-kategoriaan.

Kun toimivat säännöt oltiin otettu käyttöön Sourcefiressä, ajettiin sääntöpolitiikka uudestaan IPS-sensorille, jolloin se tulee käyttöön. Tämän jälkeen käynnistettiin uusi NTOSpider-skannaus web-sovellusta kohden. Odotettuna tuloksena oli se, että haavoittuvuuksien määrän pitäisi tippua pienemmäksi, koska nyt Sourcefirella on säännöt kyseisiä haavoittuvuuksia varten.

Kolmannen skannauksen valmistumisen jälkeen siitä muodostettu raportti käytiin jälleen läpi. Tällä kertaa haavoittuvuuksien määrä oli vain 17. Näistä XSS-haavoittuvuuksien määrä oli 10. Määrä saatiin siis puolitettua, kun NTODefendin luomat säännöt otettiin käyttöön IPS:ssä. Korkeimman prioriteetin haavoittuvuutta SQL-injektiota vastaan ei saatu paikattua, koska NTODefendin luoma Snort-sääntö ei noudattanut validia syntaksia. Tulokset olivat muuten erittäin positiiviset.



Kuva 27. Viimeisen skannauksen tulokset. Kuvasta näkee, kuinka XSS-haavoittuvuudet ovat puolittuneet.

6.4 Johtopäätökset

DAST-järjestelmän suurin hyöty tulee sen yleisestä raportista, joka kertoo hyvin web-sovelluksen tietoturvatasosta. Tässä tapauksessa NTOSpiderin luomat raportit olivat hyvin kattavia ja niiden kattavat graafiset taulukot kertoivat nopeasti tietoturvan yleisen tilanteen. Web-sovelluksen tietoturvaongelmien korjauksen kustannusarviot eivät ole luotettavia Suomalaisesta näkökulmasta, koska työvoima on erihintaista, mutta tuo kuitenkin omalla tavallaan hienon lisän raporttiin. NTOSpiderillä suoritettavaa skannausta voi suositella erittäin kriittisille web-sovelluksille, kun halutaan nähdä mahdolliset ohjelmointivirheet itse web-sovelluksessa ja siihen liitetyssä tietokannassa.

IPS-järjestelmänä käytetty Sourcefire pystyy turvaamaan web-sovelluksen osittain. Tämä vaatii kuitenkin sen, että IPS:n hallinoija tietää web-sovelluksen toimintaperiaatteen ja ne haavoittuvuudet, mitä vastaan web-sovellus täytyy suojata. Snort-sääntöjen käsin kirjoittaminen ei ole optimaalista ja vaikka laitevalmistajalta tuleekin kattava sääntökanta, ei se tarjoa täydellistä suojausta web-sovellukselle.

NTODefendillä tehdyt säännöt, jotka otettiin käyttöön Sourcefiressä, tuovat lisäturvaa web-sovelluksen tietoturvaa ajatellen. Tuloksista huomattiin, että esimerkiksi XSS-

haavoittuvuudet saatiin puolitettua tällä tavalla. NTODefend myös mahdollistaa IPS-järjestelmän hallinnoijan suojaamaan web-sovellusta luomalla Snort-sääntöjä, joka muuten olisi hyvin hankalaa. Valitettavasti NTODefend ei kuitenkaan toiminut täysin oikein ja SQL-injektiota käsittelevää sääntöä ei saatu siirrettyä sellaisenaan IPS-järjestelmään. NTODefendin yleinen käyttö on kuitenkin hyvin helppoa ja nopeaa ja se mahdollistaa web-sovelluksen nopean suojauksen, kun sitä tarvitaan.

Lähteet

1 Nations, Daniel. Web applications. Verkkodokumentti. Viitattu 15.1.2013. Saatavissa: http://webtrends.about.com/od/webapplications/a/web_application.htm.

2 Medical Databases & Web Engineering. Web Applications. Verkkodokumentti. Viitattu 15.1.2013. Saatavissa: http://www.meddb.be/webapplications.aspx.

3 Peterson, Jeremy. "Benefits of using the n-tiered approach for web applications". Verkkodokumentti. Viitattu 15.1.2013. Saatavissa: http://dc436.4shared.com/doc/2vKEMPNi/preview.html.

4 Apache HTTP palvelin. Apache. Verkkodokumentti. Viitattu 15.1.2013. Saatavissa: http://httpd.apache.org/.

5 Microsoft.com. IIS. Verkkodokumentti. Viitattu 15.1.2013. Saatavissa: http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/ddf1d92f-3e6e-423f-b024-35cefc10a22f.mspx?mfr=true.

6 IETF. Hypertext Transfer Protocol – HTTP/1.1. Verkkodokumentti. Viitattu 20.1.2013. Saatavissa: http://tools.ietf.org/html/rfc2616.

7 IETF. HTTP State Management Mechanism. Verkkodokumentti. Viitattu 20.1.2013. Saatavissa: http://tools.ietf.org/html/rfc6265#section-.

8 Heitmeyer, David. HTTP Cookies. Verkkodokumentti. Viitattu 20.1.2013. Saatavissa: http://cscie12.dce.harvard.edu/lecture_notes/2011/20110504/handout.html.

9 Microsoft.com. How TLS/SSL Works. Verkkodokumentti. Viitattu 20.1.2013. Saatavissa: http://technet.microsoft.com/en-us/library/cc783349(v=ws.10).aspx.

10 IETF. HTTP Over TLS. Verkkodokumentti. Viitattu 20.1.2013. Saatavissa: http://www.ietf.org/rfc/rfc2818.txt.

11 W3.org. Client-side Scripting and HTML. Verkkodokumentti. Viitattu 20.1.2013. Saatavissa: http://www.w3.org/TR/WD-script-970314.

12 Bradley, Angelina. Server Side Scripting. Verkkodokumentti. Viitattu 25.1.2013. Saatavissa: http://php.about.com/od/programingglossary/g/server_side.htm.

13 Whitaker, Andrew; Newman, Daniel. Penetration Testing and Network Defense. Oppikirja. Viitattu 25.1.2013.

14 Hakala, Jani. Keskustelu. 25.1.2013.

15 Owasp.org. OWASP Top 10 Application Security Risks – 2010. Verkkodokumentti. Viitattu 25.1.2013.
Saatavissa: https://www.owasp.org/index.php/Top_10_2010-Main.

16 Dr. Alkharobi, Talal. Firewalls. Verkkodokumentti. Viitattu 1.2.2013.
Saatavissa: http://www.ccse.kfupm.edu.sa/~talal/Sec/Firewall.pdf.

17 Wikipedia. Application Firewall. Verkkodokumentti. Viitattu 1.2.2013.
Saatavissa: http://en.wikipedia.org/wiki/Application_firewall.

18 Lipson, Howard; van Wyk, Ken. Application Firewalls and Proxies – Introduction and Concept of Operations. Verkkodokumentti. Viitattu 1.2.2013.
Saatavissa: https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/assembly/30-BSI.html.

19 Mitchell, Bradley. Visual Networking Overview – The OSI Model. Verkkodokumentti. Viitattu 1.2.2013.
Saatavissa:
http://compnetworking.about.com/od/basicnetworkingconcepts/l/blbasics_osimod.htm.

20 Scarfone, Karen; Mell, Peter. Guide to Intrusion Detection and Prevention Systems (IDPS). Verkkodokumentti. Viitattu 1.2.2013.
Saatavissa: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf.

21 McMillan, Jim. "Intrusion Detection FAQ: What is the difference between an IPS and a Web Application Firewall?". Verkkodokumentti. Viitattu 1.2.2013.
Saatavissa: http://www.sans.org/security-resources/idfaq/ips-web-app-firewall.php.

22 Eng, Chris. A Dose of Reality on Automated Static-Dynamic Hybrid Analysis. Verkkodokumentti. Viitattu 1.2.2013.
Saatavissa: http://software-security.sans.org/downloads/appsec-2011-files/veracode-a-dose-of-reality-on-hybrid-analysis.pdf.

23 Los, Rafal. Dynamic Application Security Testing (DAST) – When Not to Let Automation Drive. Verkkodokumentti. Viitattu 1.2.2013.
Saatavissa: http://h30499.www3.hp.com/t5/Following-the-Wh1t3-Rabbit/Dynamic-Application-Security-Testing-DAST-When-Not-to-Let/bap/5338259.